SYBASE®

XML Modeling

Sybase® PowerDesigner®

15.0

Windows

Part number: DC20014-01-1500-01
Last modified: September 2008

Contents

About This Manual

Subject

This book describes the PowerDesigner XML Model, including how to create an XSM, build each of the available diagrams, and generate and reverse engineer XML schemas.

Audience

This book assumes that you are an experienced Windows user with some experience of XML.

Documentation primer

For information about the complete documentation set provided with PowerDesigner, see the "Getting Started with PowerDesigner" chapter of the *Core Features Guide* .

Typographic conventions

PowerDesigner documentation uses special typefaces to help you readily identify specific items:

♦ `monospace text (normal and bold)`

  Used for: Code samples, commands, compiled functions and files, references to variables.

  Example: `declare user_defined...`, the `BeforeInsertTrigger` template.

♦ **bold text**

  Used for: New terms.

  Example: A **shortcut** has a target object.

♦ SMALL CAPS

  Used for: Key names.

  Example: Press the ENTER key.

Bibliography

W3C XML Recommendation – http://www.w3.org/TR/REC-xml

W3C DTD Recommendation – http://www.w3.org/TR/REC-xml#dt-doctype

W3C XML Schema Recommendation –
http://www.w3.org/XML/Schema#dev

W3C XML-Data Note –
http://www.w3.org/TR/1998/NOTE-XML-data-0105/

CHAPTER 1

Getting Started with XML Modeling

About this chapter

This chapter presents the PowerDesigner XML Model, and describes how to create your model.

Contents

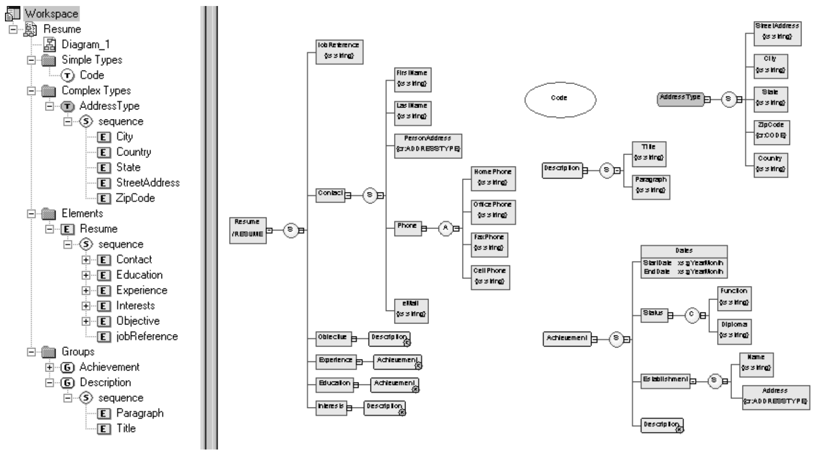
XML Modeling with PowerDesigner

An XML model (XSM) is a graphical representation of an XML Schema Definition file (.XSD), a Document Type Definition file (.DTD) or an XML-Data Reduced file (.XDR).

XML (or eXtensible Markup Language) is increasingly used to hold application data because it:

- ♦ describes and structures data, whereas HTML only displays data
- ♦ uses a self-describing and personalized syntax
- ♦ can be exchanged between incompatible systems, since data is stored in plain text format

Since XML structures can be very complex, it is much easier to visualize them through comprehensive and explicit diagrams, than to read XML-coded pages. With its Browser tree view and diagram, a PowerDesigner XSM gives you a global and schematic view of all the elements composing your XSD, DTD, or XDR:

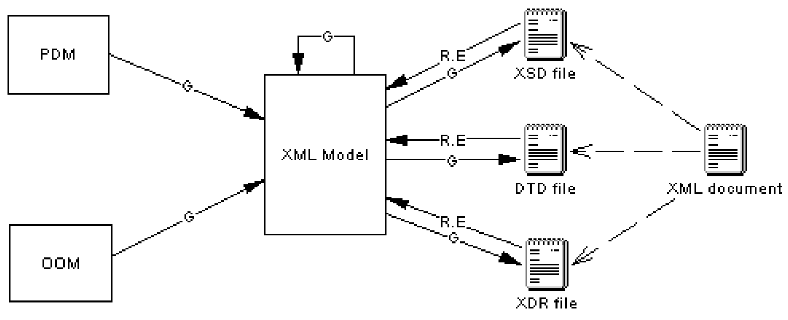


Once you have created an XML diagram, you can generate an XSD, a DTD or an XDR file from it for use in your application.

A PowerDesigner XSM allows you to:

- ♦ Build and check XML models
- ♦ Map objects in, and create reports of XML models
- ♦ Generate and reverse engineer XSD, DTD and XDR files

♦ Generate an XML model from a Physical Data Model (PDM), Object Oriented Model (OOM), or another XSM



R.E: Reverse Engineering
G: Generation

DTD, XSD or XDR

The structure of an XSM is described by a DTD, an XSD or an XDR file:

♦ A DTD file is a basic way to describe the structure of an XML document. It is a raw list of all the legal elements making up an XML document. An extract of a DTD file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Project Management -->

<!ELEMENT Database (DIVISION,EMPLOYEE,CUSTOMER,PROJECT,
TEAM,MATERIAL,PARTICIPATE,MEMBER,USED,COMPOSE)>
<!ELEMENT DIVISION EMPTY>
<!ATTLIST DIVISION
           DIVNUM                        CDATA
           DIVNAME                       CDATA
           DIVADDR                       CDATA>
<!ELEMENT EMPLOYEE EMPTY>
<!ATTLIST EMPLOYEE
           EMPNUM                        CDATA
           EMP_EMPNUM                    CDATA
           DIVNUM                        CDATA
           EMPFNAM                       CDATA
           EMPLNAM                       CDATA
           EMPFUNC                       CDATA
           EMPSAL                        CDATA>
```

♦ An XSD file (or schema) is an elaborated way to describe the structure of an XML document. It can support namespaces, derivations, keys, simple and complex user-defined data types and a robust collection of predefined data types. An extract of an XSD file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Project Management
-->
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Database">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="DIVISION">
               <xs:complexType>
                  <xs:attribute name="DIVNUM">
                     <xs:simpleType>
                       <xs:restriction base="ID">
                          <xs:minInclusive value="1"/>
                          <xs:pattern value="00000"/>
                       </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
                  <xs:attribute name="DIVNAME" type="NAME">
                  </xs:attribute>
                  <xs:attribute name="DIVADDR" type="SHORT_TEXT">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
```

An XSD file always starts with the <schema> tag (root element). All objects created in the model will appear in the XSD file between the schema start-tag and end-tag

- ♦ An XDR file is a simplified XSD file (or schema). It does not support simple and complex user-defined data types. An extract of an XDR file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="PROJECT"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <description>Project Management</description>
  <ElementType name="DIVISION" content="empty">
     <AttributeType name="DIVNUM"/>
     <attribute type="DIVNUM"/>
     <AttributeType name="DIVNAME" dt:type="string"/>
     <attribute type="DIVNAME"/>
     <AttributeType name="DIVADDR" dt:type="string"/>
     <attribute type="DIVADDR"/>
  </ElementType>
```

An XDR file always starts with the <schema> tag (root element). All objects created in the model will appear in the XDR file between the schema start-tag and end-tag

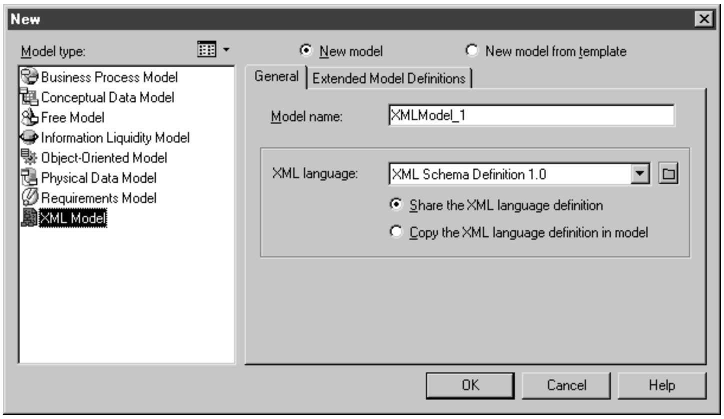
Creating an XSM

You can create a new XSM from scratch, or reverse engineer the model from an existing XSD, a DTD or an XDR file.

For information about reverse engineering, see the Generating and reverse engineering an XSD, a DTD or an XDR file chapter.

❖ To create a new XSM

1. Select File ➤ New to display the New dialog box.



[✓ New model] New model from template

Model name: XMLModel_1

XML language: XML Schema Definition 1.0

[✓ Share the XML language definition]

Copy the XML language definition in model

2. Select XML Model in the list of model types.

3. Select one of the following radio buttons:
   - New model – Creates a new, standard, model.
   - New model from template – Creates a model from a model template. A model template is a set of model options, display preferences, extension, or objects saved in a model located in the template folder. You use model templates when you need to reuse preferences and options in several models.

     ☞ For more information about model templates, see the “Model templates” section in the Models chapter of the *Core Features Guide* .

4. Type a model name in the Model name box. The code of the model, which may be used for script or code generation, is derived from this name according to the model naming conventions.

5. Select an XML language from the list.

   XML languages are defined in dedicated XML files (with a .xsl extension), which are provided as part of your PowerDesigner installation in the "\Resource Files\XML languages" directory, and contain all the syntax and specifications for each target XML language.

6. Select one of the following radio buttons:
   - ♦ Share the XML language definition – use the original XSM file in the "Resource Files\XML languages" directory. Any changes made to the XML language are shared by all linked XSMs.
   - ♦ Copy the XML language definition in model – make a copy of the original XML language file in the "Resource Files\XML languages" directory. The current XML language is independent of the original XML language, so modifications made to the file in the XML languages directory are not available to the XSM. The copied object language is saved with the XSM and cannot be used without it.

     ☞ For more information on XML language properties and customizing an XML language, see "Resource File Reference" and "Working with the Resource Editor" in the Resource Files and the Public Metamodel chapter of the *Customizing and Extending PowerDesigner* manual.

7. [optional] If you want to attach one or more extended model definitions to complement the selected XML language, click the Extended Model Definitions tab, and select the extended model definitions of your choice.

   ☞ For more information on attaching extended model definition to a model, see "Extended Model Definitions" in the Resource Files and the Public Metamodel chapter of the *Customizing and Extending PowerDesigner* manual.

8. Click OK to create the new XSM in the current Workspace.

Demo example
Several example XSMs are available in the Examples directory.

XML model properties

The Model property sheet displays the definition of the current model. From this property sheet you can modify the model definition.

The General tab contains the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users

Property	Description
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the model
File name	Location of the model file. This box is empty if the model has never been saved
Author	Author of the model. You can insert a name, a space or nothing. If you insert a space, the Author field in the title box remains empty. If you intentionally leave the box empty, the Author field in the title box displays the user name from the Version Info tab of the model property sheet
Version	Version of the model. You can use this box to display the repository version or a user-defined version of the model. This parameter is defined in the display preferences of the Title node
XML language	Current XML language for the model
Default diagram	Diagram displayed by default when opening the model

XML model property sheet Detail tab

The Detail tab is only available in a model of an XSD. It contains the following properties:

Property	Description
Target Names-pace	Namespace of all the model objects. Its name is a URI which does not refer to any file but only to an assigned name. A prefix can be assigned to the namespace. All the schema elements with this prefix in their start-tag will be associated with the namespace. For example: http://www.mycompany.-com/myproduct/XMLmodel
Language	Indicator of the language used in the model. For example: en, en-GB, en-US, de, fr
ID	ID of the model. Its value must be of type ID and unique within the file containing the model. For example: XMOD1
Element Form	Form of the elements declared in the target namespace. If you select **Qualified**, elements must be qualified with the namespace prefix. If you select **Unqualified**, elements are not required to be qualified with the namespace prefix. The value of Element Form is the global default value for all the elements declared in the target namespace. To override this setting, individual elements can use the Form attribute
Attribute Form	Form of the attributes declared in the target namespace. If you select **Qualified**, attributes must be qualified with the namespace prefix. If you select **Unqualified**, attributes are not required to be qualified with the namespace prefix. The value of Attribute Form is the global default value for all the attributes declared in the target namespace. To override this setting, individual attributes can use the Form attribute
Block	Default value for the Block property of elements and complex types in the target namespace. The Block property prevents an element or a complex type with a specified type of derivation from being used in place of the inherited element or complex type
Final	Default value for the Final property of elements, simple types and complex types in the target namespace. The Final property prevents the specified type of derivation for an element, a simple type or a complex type

☞ For more information on elements, attributes, simple and complex types, see chapter Building an XML model.

XML model property sheet Items tab

The Items tab lists the model's global objects (which have no parent symbol in the diagram, and are directly linked to the <schema> tag).

The following tools are available on this tab:

Tool	Description
	Add Element - Adds an element to the model
	Add Group - Adds a group of elements to the model
	Add Attribute - Adds an attribute to the model
	Add Attribute Group - Adds a group of attributes to the model
	Add Simple Type [XSD only]- Adds a simple type to the model.
	Add Complex Type [XSD only]- Adds a complex type to the model.
	Add Notation - Adds a notation to the model, to describe the format of non-XML data
	Add Annotation [XSD only]- Adds an annotation to the model, to provide documentation or application information.

The list reflects the order in which global objects are declared in the schema. You can change the order of declaration by selecting an item in the list and using the arrowed buttons, at the bottom-left corner of the tab, to move it in the list

☞ For more information on these items, see chapter Building an XML model.

XML model property sheet External Schemas tab

The External Schemas tab is only available in a model of an XSD.

This tab allows you to link to and reuse in your model global objects from other schemas. The following tools are available on this tab:

Tool	Description
	Add Include - Adds a specified schema file to be included in the target namespace of the current schema.
	Add Import - Adds a specified namespace whose schema components are referenced by the current schema.
	Add Redefine - Adds a specified schema file whose simple and complex types, groups and attribute groups can be redefined in the current schema.
	Add Annotation - Adds an annotation to the model to provide documentation or application information.

☞ For more information on these items, see chapter Building an XML model.

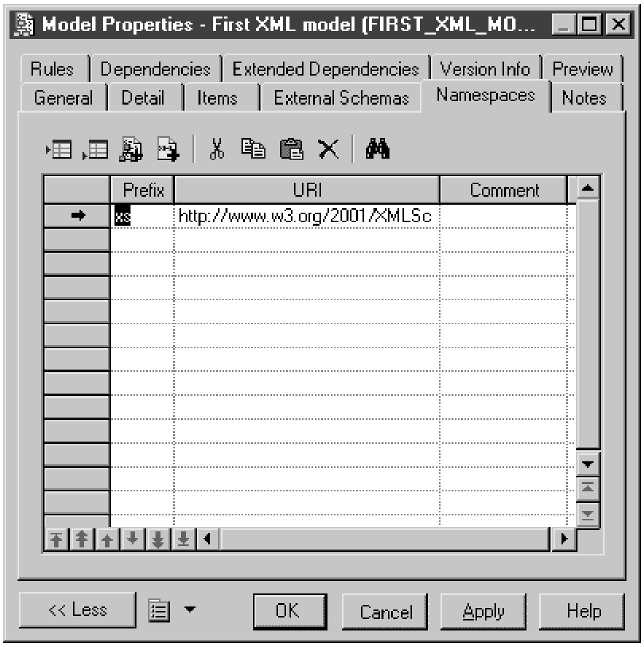
XML model property sheet Namespaces tab

A namespace is a URI indicating a location where objects are declared. The prefix of a namespace, followed by a colon (:) and the name of an object, indicates that this object is declared in that namespace. Namespaces are not supported by DTDs.

The following tools are available on this tab:

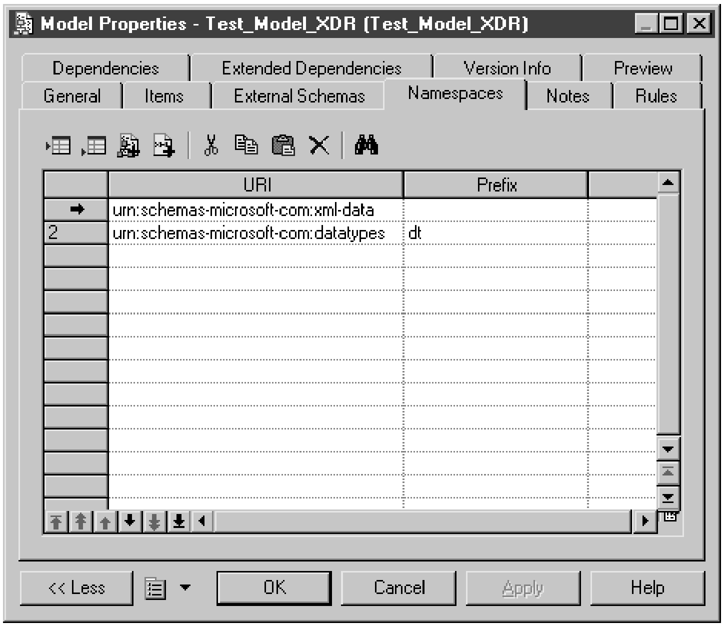
Tool	Description
	Add Namespaces from XML Models - Adds a namespace from another XSM as a source namespace for the current model.
	Add Namespaces from XML schema files - Adds a namespace from an external schema file as a source namespace for the current model.

♦ In the case of a model targeted with XSD, the namespace of the W3C XML Schema Recommendation is predefined in the list of namespaces.



	Prefix	URI	Comment
➡	xs	http://www.w3.org/2001/XMLSc	

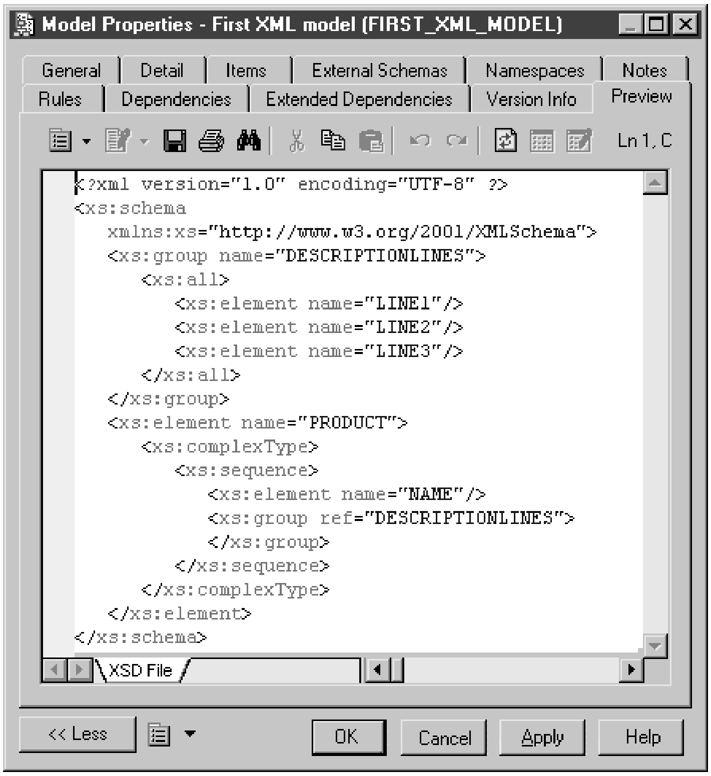
♦ In the case of a model targeted with XDR, two namespaces are predefined in the list of namespaces.



	URI	Prefix
➡	urn:schemas-microsoft-com:xml-data	
2	urn:schemas-microsoft-com:datatypes	dt

XML model property sheet Preview tab

The Preview tab of the model property sheet displays a preview of the XSD, DTD or XDR file generated from the XML model. For example:



```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:group name="DESCRIPTIONLINES">
      <xs:all>
         <xs:element name="LINE1"/>
         <xs:element name="LINE2"/>
         <xs:element name="LINE3"/>
      </xs:all>
   </xs:group>
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="NAME"/>
            <xs:group ref="DESCRIPTIONLINES">
            </xs:group>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

The schema file starts with the XML declaration followed by the <schema> (root element) declaration.

All objects created in the model will appear in the schema file between the schema start and end tags.

Customizing the XSM Environment

The XML model environment includes a set of parameters and configuration options that define various aspects of the model content and behavior. You can set these parameters:

♦ At model creation

♦ After creating a model with default options and parameters

♦ When creating a model template

Working with XML languages

An XML language contains specifications for a particular language. It provides PowerDesigner with the syntax and guidelines for implementing stereotypes, data types, scripts and constants for an XML language. You manage an XML language from the Resource Editor. The language displays a tree view with several categories that can be used to extend XML model objects (Profile category) or manage generation (Generation category).

Each XML model is by default attached to an XML language. When you create a new XML model, you must choose an XML language. You can create a new XML language or use the XML languages delivered with PowerDesigner.

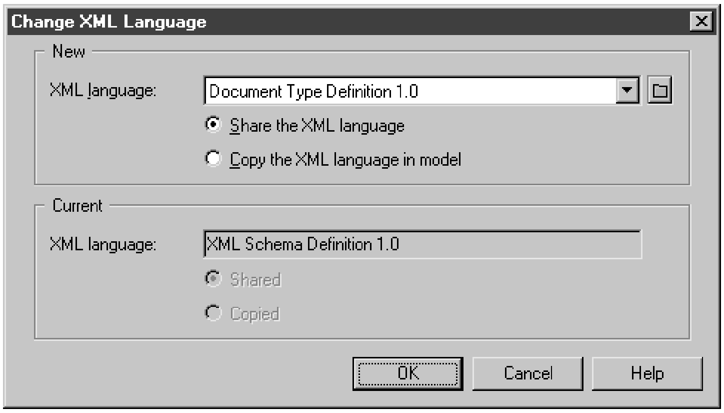
The definition of an XML language is available from its property sheet. You can select and configure parameters used when defining objects or generating from an XML model.

☞ For more information on resource files, “Resource File Reference” in the Resource Files and the Public Metamodel chapter of the *Customizing and Extending PowerDesigner* manual.

Not certified resource file
Some resource files are delivered with “Not Certified” in their names. Sybase will perform all possible validation checks, however Sybase does not maintain specific environments to fully certify these resource files. Sybase will support the definition by accepting bug reports and will provide fixes as per standard policy, with the exception that there will be no final environmental validation of the fix. Users are invited to assist Sybase by testing fixes of the definition provided by Sybase and report any continuing inconsistencies.

❖ To change the XML language of an XML model

1. Select Language ➤ Change Current Language to open the Change XML Language dialog box.



Change XML Language

New

XML language: Document Type Definition 1.0

(●) Share the XML language

( ) Copy the XML language in model

Current

XML language: XML Schema Definition 1.0

(●) Shared

( ) Copied

2. Select the XML language you want to model.

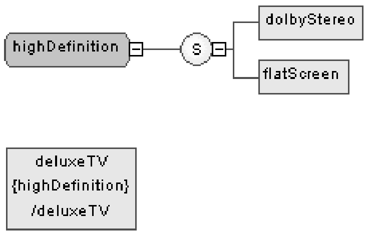
3. Select one of the following radio buttons:
   - ♦ Share the XML language – use the original XSM file in the “Resource Files\XML languages” directory. Any changes made to the XML language are shared by all linked XSMs.
   - ♦ Copy the XML language definition in model – make a copy of the original XML language file in the “Resource Files\XML languages” directory. The copied object language is saved with the XSM and cannot be used without it.

4. Click OK to change the XML language of the model.

Changes concerning simple and complex types

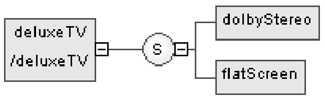
Simple types and complex types are only supported by XSDs (schemas). When changing an XSD into a DTD or an XDR, simple types and global complex types (directly linked to the <schema> tag) disappear from the diagram and the Browser tree view. Local complex types (within an element) are expanded in the diagram, beneath their containing element.

- ♦ Example of a complex type with XSD:



HighDefinition is a global complex type, reused as data type for the deluxeTV element.

- The same example with DTD or XDR:



☞ For more information on simple and complex types, see sections Defining simple types and Defining complex types in chapter Building an XML model.

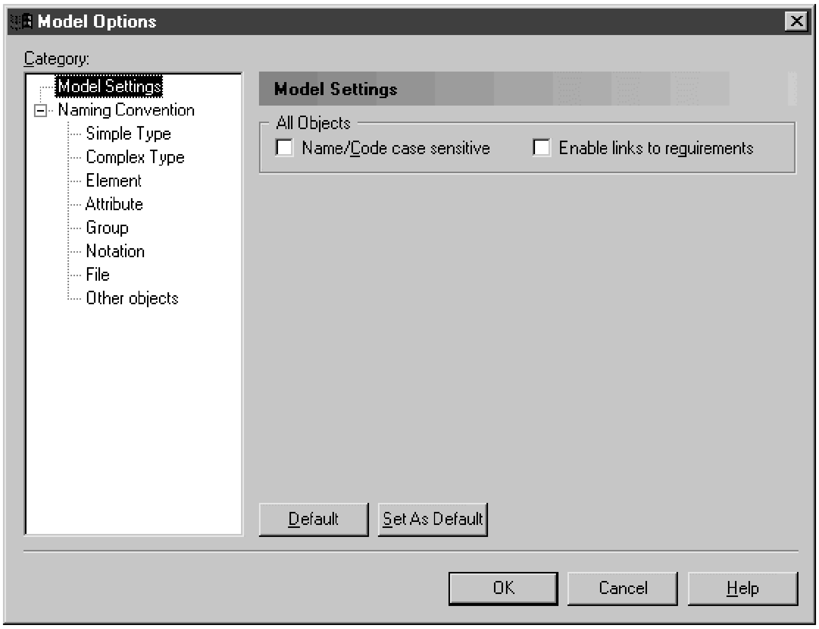
Setting XSM model options

This section explains how to set global options for the objects in your XSM. These options apply only to the current XSM.

For information about controlling the naming conventions of your models, see “Naming Conventions” section in the Models chapter of the *Core Features Guide* .

Setting Model Settings

To set Model Settings, select Tools ➤ Model Options or right-click the diagram background and select Model Options from the contextual menu.



The options on this tab affect all the objects in the model, including those already created, while changes to the object-specific options on the sub-category tabs only affect objects created subsequently.

You can set the following options on this tab:

Option	Description
Name/Code case sensitive	You can define the case sensitivity of names and codes for all objects in the current model. When this check box is selected, it implies that you can have two objects with identical name or code but different case in the same namespace. Unlike other model options, you can modify the name and code case sensitivity during the design process. However, if you do so, make sure you run the check model feature to verify if the model does not contain any duplicate object.
Enable links to requirements	Requirements are descriptions of customer needs that must be satisfied during development processes. You can enable links to requirements for all objects in the current model. When this check box is selected, it implies that the **Requirements** tab is displayed in the objects property sheet. The Requirements tab allows you to attach requirements to objects; these requirements are defined in the Requirements models open in the workspace. Attached requirements and Requirements models are synchronized. For more information on requirements, see the *Requirements* Modeling guide.

Setting XSM Display Preferences

PowerDesigner display preferences allow you to customize the format of object symbols, and the information that is displayed on them.

To set XSM display preferences, select Tools ➤ Display Preferences or right-click the diagram background and select Display Preferences from the contextual menu.

For information about changing the format of symbols, see “Format display preferences” in the Customizing your Modeling Environment chapter of the *Core Features Guide* . The following sections list the options available to customize the information displayed on XSM object symbols. Note that the objects available to be customized in the Display Preferences window are dependant upon the current diagram type.

Element display preferences

To set display preferences for elements, select Tools ➤ Display Preferences, and select the Element sub-category in the left-hand Category pane.

Preference	Description
Attributes	Displays attributes and attribute values of the element
Display limit	Maximum number of attributes displayed
Stereotype	Displays the stereotype of the element
Type	Displays the data type of the element
Comment	Displays the comment of the element
Show XPath	An XPath expression indicates the relationship between an element and the root element (<schema> tag). Select **Never**, if you do not want the elements XPath expressions to be displayed. Select **Always**, if you want all the elements to have their XPath expression displayed. Select **Root symbol**, if you only want the root symbols (global elements in the main diagram or parent elements in partial diagrams) to have their XPath expression displayed
Element Attributes	Select **Type**, if you want the attributes data types to be displayed

Complex type display preferences

To set display preferences for complex types, select Tools ➤ Display Preferences, and select the Complex Type sub-category in the left-hand Category pane.

Preference	Description
Attributes	Displays attributes and attribute values of the complex type
Display limit	Maximum number of attributes displayed
Stereotype	Displays the stereotype of the complex type
Type	Displays the data type of the complex type
Comment	Displays the comment of the complex type
Complex Type Attributes	Select Type, if you want the complex type attributes data types to be displayed

Group, simple type, and any display preferences

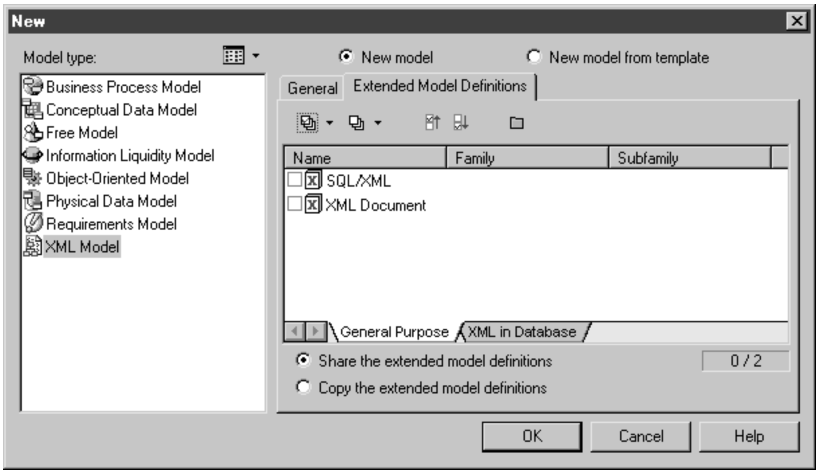
To set display preferences for groups, simple types, and anys, select Tools ➤ Display Preferences, and select the appropriate sub-category in the left-hand Category pane.

Preference	Description
Stereotype	Displays the stereotype of the group
Comment	[groups only] Displays the comment of the group

Working with XSM extended model definitions

An extended model definition allows you to expand object definitions and complement the generation targets and commands. Extended model definitions are created and saved in files with the XEM extension. You can create or attach one or several extended model definitions to a model.

When you create a new XML model, or when you reverse engineer into a new XML model, you can select one or several extended model definitions and attach them to the model from the New dialog box.



[✓ New model] New model from template

Name	Family	Subfamily
☐ SQL/XML		
☐ XML Document		

General Purpose | XML in Database

[✓ Share the extended model definitions] 0 / 2

Copy the extended model definitions

☞ For more information on extended model definitions, see "Extended Model Definitions" in the Resource Files and the Public Metamodel chapter of the *Customizing and Extending PowerDesigner* manual.

Working with XSM extended dependencies

Extended dependencies are links between objects of an XML model. These links help to make object relationships clearer but are not interpreted and checked by PowerDesigner as they are meant to be used for documentation purposes only.

You can complement these links by applying stereotypes. Stereotypes can be used to define extended dependencies between objects in an XML model.

You can type stereotypes directly in the Stereotype column of the object property sheet or select a value from the list if you have previously defined stereotypes in an embedded or imported extended model definition (.XEM).

☞ For more information on extended model definitions, see “Extended Model Definitions” in the Resource Files and the Public Metamodel chapter of the *Customizing and Extending PowerDesigner* manual.

CHAPTER 2

Building XML Models

About this chapter

This chapter describes how to build an XML model (XSM). It explains the role of each object in an XML model and how to create and modify them.

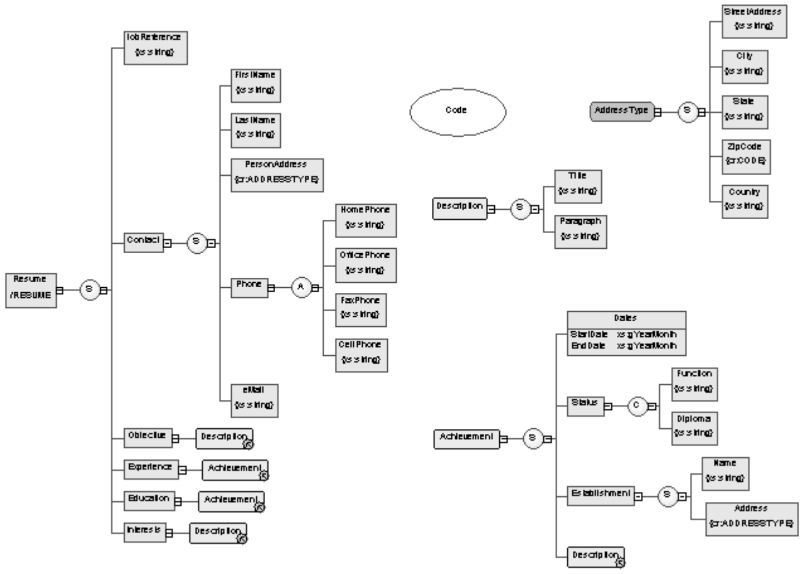
Contents

XML Diagram Basics

An XML diagram is the easiest way to define the structure and content of an XML document if you are not familiar with the syntax of XML Schema Definition (XSD), Document Type Definition (DTD) or XML-Data Reduced (XDR).

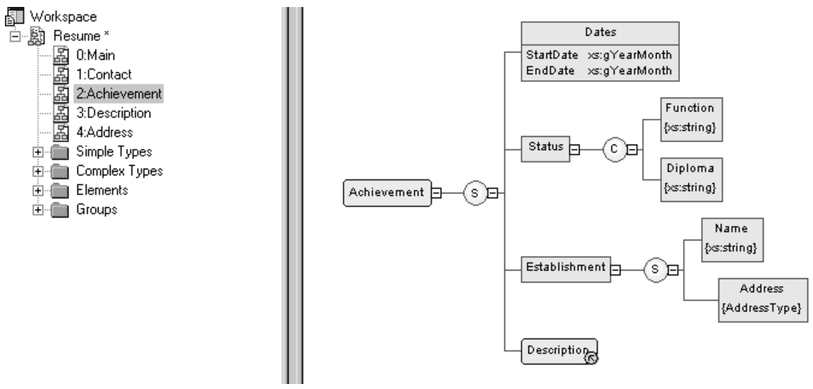
With the user-friendly graphical interface of PowerDesigner XML Model, you can build an XML diagram and then generate automatically an XSD, a DTD or an XDR file.

The following example shows the diagram of an XSM which models an XML schema for Resume documents:



If an XML model is too large or too complex, you can create several diagrams to have partial views of the model and focus on certain objects.

For example, the original Resume diagram could be split into five diagrams, corresponding to the five main objects of the model (Main, Contact, Achievement, Description and Address). The Achievement sub-diagram follows:



XML diagram objects

An XML model represents the structure of a potential or existing XSD, DTD, or XDR through a tree structure of child elements attached to parent elements.

Elements are the basic describing items of an XML model. They can be made of other elements combined in different ways through group particles. Elements are specified by attributes and data types, which can be predefined or user-defined. Simple and complex data types can be defined as global (directly linked to the <schema> tag) or local (embedded in an element declaration).

You can create the following objects in an XML diagram:

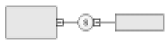
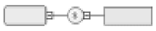
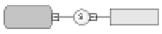
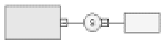
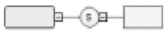
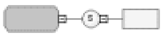
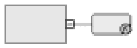
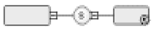
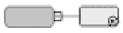
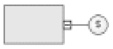
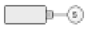
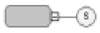
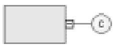
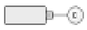
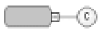
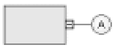
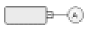
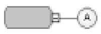
Object	Tool	Symbol	Description
Element	<E>	Element /Element	The basic object of an XML model. An element can contain other elements or attributes. See "Elements (XSM)" on page 29.
Any	a	Any	Any type of object. Can only be attached to a sequence or a choice group particle. See "Any Elements (XSM)" on page 44.
Attribute	N/A	N/A	Additional information about an element or a complex type. Defined by a built-in data type or a simple data type. See "Attributes (XSM)" on page 46.

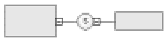
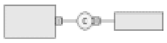
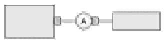
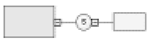
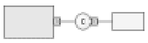
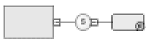
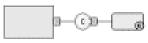
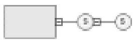
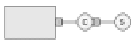
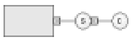
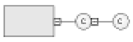
Object	**Tool**	**Symbol**	**Description**
Group	G	Group	A group of elements arranged by a group particle. Defined once and reused through references. See “Groups (XSM)” on page 62.
Attribute Group	N/A	N/A	A group of attributes, defined once and reused in the model through references. See “Attribute Groups (XSM)” on page 68.
Simple Type	N/A	N/A	[XSD only] Used in the case of elements or attributes with text-only content. See “Simple Types (XSM)” on page 71.
Complex Type	T	Complex	[XSD only] Used to introduce elements or attributes within an element declaration. See “Complex Types (XSM)” on page 73.
Sequence	S	S	This group particle arranges a set of elements, where all the elements must appear at least once in the order of their declaration. See “Group Particles (XSM)” on page 39.
Choice	C	C	This group particle arranges a set of elements, from which one element must be chosen. See “Group Particles (XSM)” on page 39.
All	A	A	This group particle arranges a set of elements, where each element can appear or not, in any order. See “Group Particles (XSM)” on page 39.
Instruction	N/A	N/A	An import, include, or redefine instruction. See “Instructions: Import, Include and Redefine (XSM)” on page 96
Derivation	N/A	N/A	Extends or restricts the values of elements and simple and complex types. See “Derivations: Extensions, Restrictions, Lists and Unions (XSM)” on page 79

Object	Tool	Symbol	Description
Constraint	N/A	N/A	[XSD only] Specifies uniqueness of element values. See "Constraints: Keys, Uniques, and KeyRefs (XSM)" on page 54
Annotation	N/A	N/A	Provides documentation or application information. See "Annotations (XSM)" on page 89
Entity			[DTD only] Specifies a predefined value or external XML or non-XML file. See "Entities (XSM)" on page 94.
Notation	N/A	N/A	Defines and processes non-XML objects within an XML model. See "Notations (XSM)" on page 92

Linking objects in an XML model

XML objects do not support standard link objects. To link a child object to a parent object, you must click the child object tool in the palette and then click the symbol of the parent object in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	**Element symbol**	**Group symbol**	**Complex type symbol**
			
Any			
			
	No link	No link	No link
			
			
All			

Tool	**Sequence symbol**	**Choice symbol**	**All symbol**
			
Any			No link
			No link
	No link	No link	No link
			No link
			No link
All	No link	No link	No link

Caution

A group particle (sequence, choice, all) cannot be created from scratch in a diagram. It must be the child element of an element, a group or a complex type.

☞ For more information, see sections How to link a child object to an element, How to link a child object to a group particle, How to link a child object to a group of elements, How to link a child object to a complex type, in chapter Building an XML model.

Creating an XML diagram

You can create an XML diagram in an existing XSM in any of the following ways:

- ♦ Right-click the model in the Browser and select New ➤ XML Model Diagram from the contextual menu
- ♦ Right-click the background of any diagram and select Diagram ➤ New Diagram ➤ New Diagram from the contextual menu.

To create a new XSM with an XML diagram, select File ➤ New, choose XML Model from the Model type list, and click OK.

Group Symbols feature

The Symbol ➤ Group Symbols feature is only available for free symbols in an XML diagram.

Expand/Expand All/Collapse/Arrange Symbols features

Right-click a symbol in an XML diagram and select one of these features in the contextual menu: **Expand**: the hierarchy below a symbol is partially expanded (only the first level). **Expand All**: the hierarchy below a symbol is fully expanded (all levels). **Collapse**: the hierarchy below a symbol is hidden. **Arrange Symbols**: the hierarchy below a symbol is properly displayed.

External Shortcuts (through references and data types)

External shortcuts allow you to share objects between different models. You can define external shortcuts in an XML model, but you cannot use them directly in the model, except as substitution groups for elements (see Detail tab in element property sheet).

You can define external shortcuts for any global object (with no parent object in the diagram), except for imports, includes, redefines and annotations.

Internal shortcuts allow you to share objects between packages of a same model. You cannot define internal shortcuts since an XML model does not support packages.

External shortcuts are automatically generated in the following situations:

References

When you use the Reference property to define an element, an attribute, a group or an attribute group, by reference to a similar object in another model opened in the workspace, a shortcut is created between the referencing object and the target object.

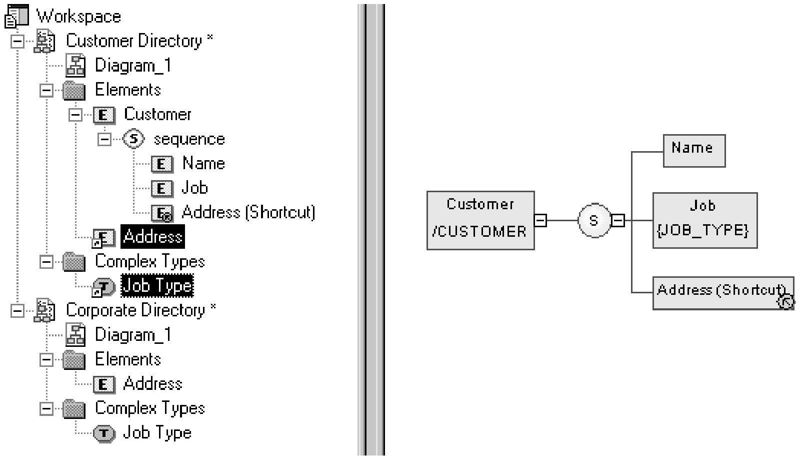
The shortcut is displayed in the current model with a specific item in the Browser tree view and the "(Shortcut)" expression in the reference symbol and item. The target object keeps track of the referencing object in the Reference tab of the Dependencies tab of its property sheet.

Data types

When you define the data type of an element by selecting a simple or a complex type from another model (using the Browse tool beside the Type list), a shortcut is created between the current element type and the target data type.

The shortcut is displayed in the current model with a specific item in the Browser tree view.

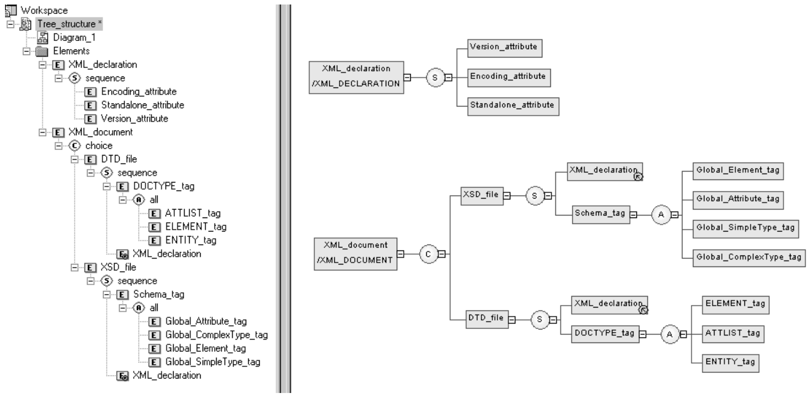
Example of shortcuts through a reference and a data type:



Elements (XSM)

Elements are the basic building blocks of an XML model.

An XML model is a tree structure of elements where child elements are attached to parent elements. For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="XML_DOCUMENT">
      <xs:complexType>
         <xs:choice>
            <xs:element name="XSD_FILE">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="XML_DECLARATION"/>
                     <xs:element name="SCHEMA_TAG">
                        <xs:complexType>
                           <xs:all>
                              <xs:element name="GLOBAL_ELEMENT_TAG"/>
                              <xs:element name="GLOBAL_ATTRIBUTE_TAG"/>
                              <xs:element name="GLOBAL_SIMPLETYPE_TAG"/>
                              <xs:element name="GLOBAL_COMPLEXTYPE_TAG"/>
                           </xs:all>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DTD_FILE">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="XML_DECLARATION"/>
                     <xs:element name="DOCTYPE_TAG">
                        <xs:complexType>
                           <xs:all>
                              <xs:element name="ELEMENT_TAG"/>
                              <xs:element name="ATTLIST_TAG"/>
                              <xs:element name="ENTITY_TAG"/>
                           </xs:all>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:choice>
      </xs:complexType>
   </xs:element>
   <xs:element name="XML_DECLARATION">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="VERSION_ATTRIBUTE"/>
            <xs:element name="ENCODING_ATTRIBUTE"/>
            <xs:element name="STANDALONE_ATTRIBUTE"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

In a schema, elements are declared with <element> tags.

There are two broad kinds of elements:

- Global elements - have no parent element in a diagram, and are directly linked to the <schema> tag (root element) in a schema. They can be reused in the model through referencing elements (See "XML_declaration" in the "Elements (XSM)" on page 29 example)
- Local elements - have a parent element in a diagram, and are unique within their parent scope. They can be defined by reference to a global element (see the Reference property in "Element properties" on page 31).

Global and local elements in XDR files
In a model targeted with the XML-Data Reduced language, local elements are first declared separately, like global elements (with the <ElementType> tag and a name attribute), then within their parent element (with the <element> tag and a type attribute).

Extract of an XDR file:

```
<ElementType name="localElement"
<ElementType name="globalElement"
   <element type="localElement"/>
</ElementType>
```

Parent elements are linked to their child elements through group particles (sequence, choice or all), which contain a group of child elements (see the Group type property in "Element properties" on page 31).

You can derive an XSD element data type to extend or restrict its values (see the Derivation property in "Element properties" on page 31).

Creating an element

You can create an element in any of the following ways:

- ♦ Use the Element tool in the diagram Palette.
- ♦ Select Model ➤ Elements to access the List of Elements, and click the Add a Row tool.
- ♦ Right-click the model or package in the Browser, and select New ➤ Element.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Element properties

You can modify an object's properties from its property sheet. To open an element property sheet, double-click its diagram symbol or its Browser entry in the Elements folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for elements.

The General tab of an XSD or DTD element property sheet displays the following properties (for XDR element properties, see the subsequent table):

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users

Property	Description
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the element
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	Name of a global element. The current element will have the same properties as the global element. The Reference property is only available for child elements. Use the list to select a global element in the current model, or the Browse tool to select a global element from any open model. If you select a global element from another model, a shortcut is created with the referencing element. When you define a reference, name and code properties are grayed. Name and code are those of the global element. Once you have referenced an element, you can locate it in the diagram by right-clicking the referencing element symbol and selecting Find Referenced Element in the contextual menu. The referenced element is displayed with handles in the diagram.
Group type	Specifies that the element has child elements, and how they are used. You can choose between: ♦ all – All children may be present. See “Group Particles (XSM)” on page 39 ♦ choice – Only one child must be present. See “Group Particles (XSM)” on page 39 ♦ group – Reference to a predefined group. See “Groups (XSM)” on page 62 ♦ sequence – All children must be present in order. See “Group Particles (XSM)” on page 39
Type	Element data type. Use the list to select a built-in data type. Use the Browse tool to select a simple or a complex type from any model opened in the current workspace. In the case of an XSD, selecting a data type will delete any group particle (and its child elements) or attribute previously defined in the element property sheet. Do not select a data type if you want to define attributes or child elements within the current element

Property	Description
Embedded type	[XSD only] Locally defined data type. It applies to the current element only. Automatically set to Complex if you define a derivation for the element data type.
Content	[XSD only] Content type of the element. If you select **Complex**, the element can have child elements. If you select **Simple**, the element cannot have child elements.
Derivation	[XSD only] Derivation method for the element data type. Used to extend or restrict the values of the element data type. When you define a derivation, the data type disappears. You must click Apply and then the Properties tool to select a base type in the derivation property sheet. See "Derivations: Extensions, Restrictions, Lists and Unions (XSM)" on page 79.

Defining elements in XDR files

In a model targeted with the XML-Data Reduced language, elements are defined by different attributes:

XDR element attribute	Description
Model	Specifies if an element can contain new local elements. Possible values are: ♦ closed – [default] ♦ open - if an "Any" element is attached to the element. See "Any Elements (XSM)" on page 44 Tab: N/A
Content	Specifies the content type. Possible values are: ♦ mixed - a group particle and a data type are defined ♦ eltOnly - a group particle is defined without a data type ♦ textOnly - a data type is defined without a group particle ♦ empty – neither group particle nor data type are defined Tab: General Field: Group type, Type

XDR element attribute	**Description**
Order	Specifies how child elements are organized within a parent element. Possible values are: ♦ seq - sequence group particle ♦ one - choice group particle ♦ many - all group particle Tab: General Field: Group type
dt:type	Specifies a data type. Tab: General Field: Type
dt:values	Specifies a list of possible element values. Tab: Values
type	[local elements only] Specifies the name of a global element as reference for the local element Tab: General Field: Reference
minOccurs	[local elements only] To specify the minimum number of occurrences for a local element. Usually set to **0** or **1** Tab: Detail Field: Minimum
maxOccurs	[local elements only] To specify the maximum number of occurrences for a local element. Usually set to **1** or ***** (unbounded) Tab: General Field: Maximum

Example of an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_elements"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="localElement" content="empty"/>
  <ElementType name="globalElement" model="closed" content="eltOnly" order="seq" dt:values="0 1 2">
     <element type="localElement" minOccurs="0" maxOccurs="*"/>
  </ElementType>
</Schema>
```

Element property sheet Detail tab

The Detail tab contains the following properties:

Property	Description
Minimum	Minimum number of times the element can occur. To specify that the element is optional, set this attribute to zero
Maximum	Maximum number of times the element can occur. For an unlimited number of times, select unbounded
Substitution group	Name of a global element for which the current element can be substituted. It must have the same type or a derived type. Its value must be a qualified name (See Glossary)
Default	Default value of the element if its content is a simple type or text-only. Enter a default value only if there is no fixed value
Fixed	Predetermined, unchangeable value of the element if its content is a simple type or text-only. Enter a fixed value only if there is no default value
Block	Property to prevent another element with the same type of derivation from being used in place of the current element
Final	Property to prevent derivation of the current element. Prohibited if the element is not a global element
Form	Form of the element. Used to specify the target namespace of the element. If you select **Qualified**, a namespace prefix is required to qualify the element. If you select **Unqualified**, a namespace prefix is not required to qualify the element
ID	ID of the element. Its value must be of type ID and unique within the model containing the element
Abstract	Property defining if the element can appear in the instance document or not. If selected, the element cannot appear in the instance document
Nillable	Property defining if the element is null or not

In the case of a model targeted with XDR, the Detail tab is only available for local elements.

Element property sheet Attributes tab

Attributes (see "Attributes (XSM)" on page 46) give additional information about an element. The Attributes tab lists the attributes and attribute groups associated with the element.

For information about the tools available on this tab for adding attributes, see

"Creating an attribute" on page 47.

Element property sheet Constraints tab

Constraints (see "Constraints: Keys, Uniques, and KeyRefs (XSM)" on page 54) allow you to indicate that element values must be unique within their specified scope. The Constraints tab lists the constraints associated with the element.

For information about the tools available on this tab for adding constraints, see "Creating a constraint" on page 56.

Element property sheet Mapping tab

Object mapping is the ability to establish a correspondence between objects belonging to heterogeneous models and diagrams.

The Mapping tab of an element property sheet allows you to map the current element and its attributes to PDM or OOM objects.

Select a data source in the Mapping for list. If it is the first time you define a mapping for an element, the Mapping for list is empty. Click the Add a Mapping for a Data Source tool and select a data source.

Element Sources tab

The Element Sources tab allows you to associate one or several PDM or OOM objects to the current element.

You can use the Add Objects tool to select objects from the PDMs or OOMs opened in the current workspace.

Attributes Mapping tab

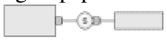
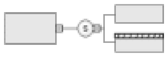
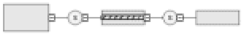
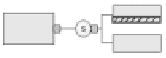
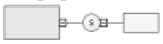
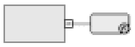
The Attributes Mapping tab allows you to define the mapping between PDM columns or OOM class attributes and the element attributes.

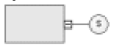
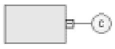
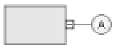
Tool	Description
	Add Mapping - Use this tool to select the attributes in the current element that will be mapped to PDM columns or OOM class attributes. Once you have selected the attributes, you can use the list in the Mapped to column to select corresponding PDM columns or OOM class attributes
	Create from Sources - Use this tool to copy PDM columns or OOM class attributes in the data source to the current element attributes
	Generate Mapping - Use this tool to automatically generate a mapping between PDM columns or OOM class attributes and element attributes with the same name or code in the data source and the current model

☞ For more information on element mapping, see section Mapping objects in an XML model in chapter Working with an XML model.

Link child objects to elements

XML objects do not support standard PowerDesigner link objects. To link a child object to an element, you must click the child object tool in the palette and then click the element symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action
→E	If you click a parent element symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet 
↑E	If you click the upper part of a child element symbol with the Element tool, a brother element symbol is displayed above the child element symbol 
→E	If you click the middle part of a child element symbol with the Element tool, a sequence group particle and a grand child element symbol are created. You can modify the group particle via its property sheet 
↓E	If you click the lower part of a child element symbol with the Element tool, a brother element symbol is displayed below the child element symbol 
A	If you click an element symbol with the **Any** tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet 
G	If you click an element symbol with the Group tool, a referencing group is created. You must now select a group for the reference 

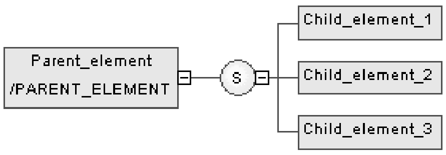
Tool	Action
⊘ Ⓣ	If you click an element symbol with the Complex Type tool, a complex type symbol is displayed superposed, but not linked, to the element symbol. A global complex type cannot be the child of an element
✛ Ⓢ	If you click an element symbol with the Sequence tool, a sequence group particle is displayed linked to the element symbol 
✛ Ⓒ	If you click an element symbol with the Choice tool, a choice group particle is displayed linked to the element symbol 
✛ Ⓐ	If you click an element symbol with the **All** tool, an all group particle is displayed linked to the element symbol 

Pointer indications
When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See elements in Tool column).

Group Particles (XSM)

An element composed of other elements is a parent element with child elements.

Child elements are linked to their parent element through a group particle.



There are three kinds of group particles:

Tool	Symbol	Description
		Sequence - Child elements must appear at least once in the order of their declaration
		Choice - Only one child element can be linked to the parent element
		All - Child elements can appear in any order and each of them once or not at all

These particles translate to the following tags in each of the supported languages:

Group Particle	XSD	XDR (order attribute)	DTD (separator)
Sequence	<sequence>	seq	, (comma)
Choice	<choice>	one	| (bar)
All	<all>	many	, (comma)

Creating a group particle

You can create a group particle in any of the following ways:

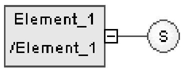
- Use the Sequence, Choice, or All tool in the diagram Palette.
- Open the property sheet of an element, group, or complex type, and select a group particle from the Group type list.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

❖ To create a group particle from the palette

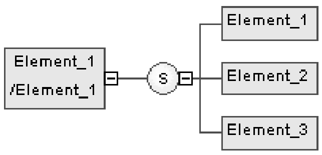
1. Select a group particle tool (Sequence, Choice, or All) in the palette, and then click the element symbol in the diagram.

   A group particle is created and its symbol is displayed in the diagram, linked to the element symbol.



2. Select the Element tool in the palette, and then click the group particle symbol to add an element item. Click the symbol again to add additional elements.

   The child elements appear one by one in the diagram, linked to the group particle symbol.



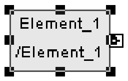
3. Right-click in order to recover the Pointer.

Clicking an element symbol with the Element tool
When you click an element symbol with the Element tool, a sequence symbol (by default) is displayed in the diagram between the parent element and the child element. To add other child elements, click the sequence symbol with the Element tool. To change the group particle, double-click the sequence symbol to display its property sheet, then select another group particle in the Type list and click OK.

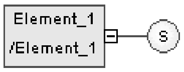
❖ To create a group particle from the property sheet of an element

1. Open the property sheet of the element, select a group particle from the Group type list, and then click OK.

   The element is displayed selected, with an **Expand tab** (+) on its right side:



2. Click an empty space in the diagram to deselect the element, and then click the Expand tab (+) to reveal the group particle symbol (in this case, a sequence):

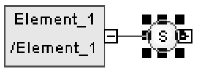
   Note that you can click the **Collapse tab** (-), on the link in order to hide the group particle.

3. Double-click the group particle to open its property sheet, and then click the Items tab.

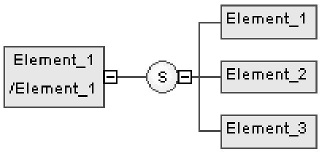
4. Click the **Add Element** tool for each child element you want to create in the list.

5. Click OK to return to the diagram.

   The group particle is displayed selected, with an Expand tab on its right side.

6. Click an empty space in the diagram, to deselect the group particle symbol, and then click the Expand tab to reveal the child element symbols.

Name and code uniqueness
Child elements are defined within the namespace of their parent element. Therefore, there cannot be a conflict between a parent and a child name.

☞ For more information on the namespace concept, see “Controlling the namespace of a package”, in the Models chapter of the *Core Features Guide* .

Group particle properties

You can modify an object’s properties from its property sheet. To open a group particle property sheet, double-click its diagram symbol or its Browser entry beneath its parent object. The following sections detail the property sheet tabs that contain the properties most commonly entered for group particles.

The General tab contains the following properties:

Property	Description
Type	Type of the group particle. You can change its type by selecting a value in the list and clicking OK
Minimum	Minimum number of times the group particle can occur. To specify that the group particle is optional, set this property to zero
Maximum	Maximum number of times the group particle can occur. For an unlimited number of times, select unbounded
ID	ID of the group particle. Its value must be of type ID and unique within the model containing the group particle

Group particle property sheet Items tab

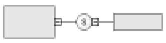
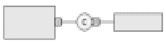
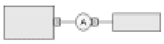
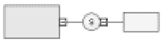
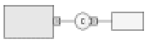
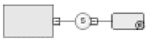
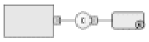
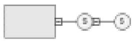
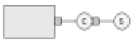
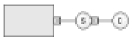
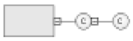
The Items tab list the child elements associated with the group particle. You can add additional children directly on this tab using the following tools:

Tool	Description
	Add Element - Adds an element to the list
	Add Any - Adds an Any to the list. Only available with a choice or a sequence group particle
	Add Group Particle - Adds a group particle to the list
	Add Reference to Element - Adds a referencing element to the list. Select a global element for the reference in the Selection dialog box. To use this tool, you must have previously defined a global element in the current model
	Add Reference to Group - Adds a referencing group to the list. Select a group for the reference in the Selection dialog box. To use this tool, you must have previously defined a group in the current model

Adding a child object to a group particle

XML objects do not support standard link objects. To link a child object to a group particle, you must click the child object tool in the palette and then click the group particle symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Caution
A group particle cannot be created from scratch in a diagram. It must be the child element of an element, a group or a complex type.

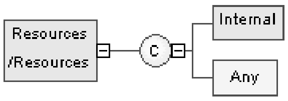
Tool	Sequence symbol	Choice symbol	All symbol
E			
A Any			No link
G	 A referencing group is created. You must now select a group for the reference	 A referencing group is created. You must now select a group for the reference	No link
T	No link	No link	No link
S			No link
C			No link
A All	No link	No link	No link

Pointer indications
When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Any Elements (XSM)

Any elements allow you to attach any type of object to a choice or a sequence group particle.

For example:



♦ In an XSD file, Any is declared with the <any> tag:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="RESOURCES">
      <xs:complexType>
         <xs:choice>
            <xs:element name="INTERNAL"/>
            <xs:any namespace="##local" processContents="lax"/>
         </xs:choice>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

♦ In a DTD file, Any is declared within an <!ELEMENT> tag with the keyword “ANY”:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT Resources ANY>
<!ELEMENT Internal EMPTY>
```

♦ In an XDR file, Any is declared through of an <ElementType> tag (resources in the example) with its **model** attribute set to “**open**”. Although it is displayed in a diagram, Any is not considered as an object in an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_Any"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="internal" content="empty"/>
  <ElementType name="resources" model="open" content="eltOnly" order="one">
     <element type="internal"/>
  </ElementType>
</Schema>
```

Creating an any element

You can create an any element in any of the following ways:

♦ Use the Any tool in the diagram Palette.

♦ Open the Items tab in the property sheet of a group particle, and click the Add Any tool.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Any element properties

You can modify an object's properties from its property sheet. To open an any element property sheet, double-click its diagram symbol or its Browser entry beneath its parent object.

The General tab contains the following properties:

Property	Description
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Minimum	Minimum number of times the Any can occur. To specify that the Any is optional, set this attribute to **zero**
Maximum	Maximum number of times the Any can occur. For an unlimited number of times, select **unbounded**
ID	ID of the Any. Its value must be of type ID and unique within the model containing the Any. Only available in a model targeted with XSD
Names-pace	Namespaces containing the objects that can be used. If you select **##any**, objects from any namespace can be used. If you select **##other**, objects from any namespace other than the target namespace of the schema can be used. If you select **##local**, objects that are not qualified with a namespace can be used. If you select **##targetNamespace**, objects from the target namespace of the schema can be used. If you type a combination of URI references, ##targetNamespace and ##local, provided they are separated by a white space, objects from this combination can be used. Only available in a model targeted with XSD
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the objects specified by the Any. If you select **Strict**, the XML processor must obtain the schema and validate any object of the specified namespaces. If you select **Lax**, the XML processor will try to obtain the schema and validate any object of the specified namespaces. If the schema cannot be found, no error will occur. If you select **Skip**, the XML processor will not try to validate the objects of the specified namespaces. Only available in a model targeted with XSD

Attributes (XSM)

Attributes are used to give additional information about elements.

There are global and local attributes:

- ♦ Global attributes are defined with the Model menu. In a schema, they are directly linked to the <schema> tag (root element). They can be reused for any element in the model through references (See “NUMBER” attribute in the generated schema)
- ♦ Local attributes only apply to the elements in which they are created. They can be defined by reference to a global attribute (See Reference property)

Global and local attributes in XDR files

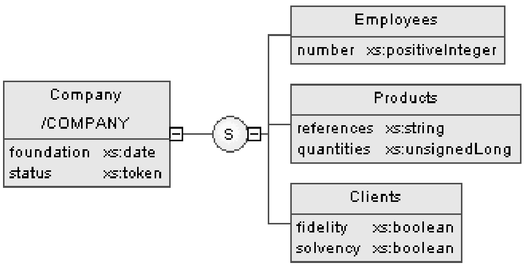
In a model targeted with the XML-Data Reduced language, local attributes are first declared separately, like global attributes (with the <AttributeType> tag and a name attribute), then within their parent element (with the <attribute> tag and a type attribute).

Extract of an XDR file:

```
<AttributeType name="globalAttribute"/>
<ElementType name="parentElement" content="empty">
   <AttributeType name="localAttribute" default="0"
   <attribute default="0" type="localAttribute"/>
</ElementType>
```

You can derive an attribute data type to extend or restrict its values. (Only with a model targeted with XSD)

For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="COMPANY">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEES">
               <xs:complexType>
                  <xs:attribute ref="NUMBER">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="PRODUCTS">
               <xs:complexType>
                  <xs:attribute name="REFERENCES" type="xs:string">
                  </xs:attribute>
                  <xs:attribute name="QUANTITIES" type="xs:unsignedLong">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="CLIENTS">
               <xs:complexType>
                  <xs:attribute name="FIDELITY" type="xs:string">
                  </xs:attribute>
                  <xs:attribute name="SOLVENCY" type="xs:string">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="FOUNDATION" type="xs:date">
         </xs:attribute>
         <xs:attribute name="STATUS" type="xs:token">
         </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:attribute name="NUMBER" type="xs:positiveInteger">
   </xs:attribute>
</xs:schema>
```

In a schema, attributes are declared with <attribute> tags.

Creating an attribute

☞ You can create attributes (and attribute groups) on the Attributes tab of an element, complex type, or attribute group using the following tools:

Tool	Description
	Add Attribute - Creates a local attribute
	Add Undefined Reference to Attribute Group - Adds an attribute group with a reference to an attribute group defined in the current model. Select a name in the Reference list. You can also type a new name in the Reference column and then define a new attribute group in the Attribute Groups list (See Model menu)
	Add Reference to Attribute - Adds one or several attributes with a reference to global attributes defined in the current model. Select one or several global attributes in the Selection dialog box
	Add Reference to Attribute Group - Adds a reference to an attribute group defined in the current model. Select one or several attribute groups in the Selection dialog box
☑	**Any Attribute** - Adds "any" attribute of a specified namespace. For more information, see ""Any" Attributes" on page 51.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Attribute properties

You can modify an object's properties from its property sheet. To open an attribute property sheet, double-click its diagram symbol or its Browser entry in the Attributes folder beneath an entity or complex type. The following sections detail the property sheet tabs that contain the properties most commonly entered for attributes.

The General tab contains the following properties:

Property	Description
Name	The name of the item which should be clear and meaningful, and should convey the item's purpose to non-technical users
Code	The technical name of the item used for generating code or scripts, which may be abbreviated, and should not generally include spaces
Comment	Descriptive label of the attribute

Property	Description
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	Name of an attribute in the current model or another model opened in the workspace. A reference allows you to reuse an attribute with all its properties without having to define it again. Use the list to select an attribute in the current model. Use the Browse tool to select an attribute from any model opened in the workspace. If you select an attribute from another model, a shortcut is created with the referencing attribute. When you define a reference, name and code properties are grayed. The name and code are those of the target attribute
Type	Attribute data type. It must be a qualified name (See Glossary). Use the list to select a built-in data type. Use the Browse tool to select a simple type defined in the current model or another model opened in the workspace
Embedded Type	If selected, the attribute data type disappears and a <simple type> tag is created in the schema within the <attribute> tag. Only available in a model targeted with XSD
Derivation	Derivation method for the attribute data type. Used to extend or restrict the values of the attribute data type. When you define a derivation, the data type disappears. You must click Apply and then the Properties tool to select a type, a base type or member types for the corresponding derivation (list, restriction or union). Only available in a model targeted with XSD

Defining attributes in XDR files

In a model targeted with the XML-Data Reduced language, attributes tags are defined by different attributes:

XDR attribute attribute	Description
name	Specifies the name of the attribute. Tab: General Field: Name
default	Specifies a default value for the attribute. Tab: Detail Field: Default

XDR attribute attribute	Description
dt:type	Specifies a type for the attribute. Tab: General Field: Type
dt:values	To specify a list of available values for a global attribute Tab: Values
type	Specifies the name of a global attribute as a reference for a local attribute. Tab: General Field: Reference

Example of an XDR file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_attributes"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <AttributeType name="globalAttribute"/>
  <ElementType name="parentElement" content="empty">
     <AttributeType name="localAttribute" default="0" dt:values="1 2 3"/>
     <attribute default="0" type="localAttribute"/>
  </ElementType>
</Schema>
```

Attribute property sheet Detail tab

The Detail tab of an attribute property sheet displays the following properties:

Property	Description
Default	Default value. Enter a default value only if there is no fixed value
Fixed	Fixed value. Enter a fixed value only if there is no default value
Use	Indicator of how the attribute is used. If you select **Optional**, the attribute is optional and may have any value. If you select **Prohibited**, the attribute cannot be used. Use this value to prohibit the use of an existing attribute in the restriction of another complex type. If you select **Required**, the attribute must appear at least once and may have any value matching its data type
Form	Form of the attribute. If you select **Qualified**, Form must be qualified by combining the target namespace of the schema with the no-colon-name (See Glossary) of the attribute. If you select **Unqualified**, Form is not required to be qualified with the namespace prefix and is matched against the no-colon-name of the attribute
ID	ID of the attribute. Its value must be of type ID and unique within the model containing the attribute

Attribute property sheet Values tab

The Values tab of an attribute property sheet is only available in a model targeted with DTD or XDR. You can set a list of predefined values for an attribute.

Element values with XDR
In a model targeted with the XML-Data Reduced language, there is also a Values tab in the element property sheet.

”Any” Attributes

The “Any” attribute feature allows you to insert any attribute of specified namespaces into an element, a complex type or an attribute group declaration. It is only available in a model targeted with XSD.

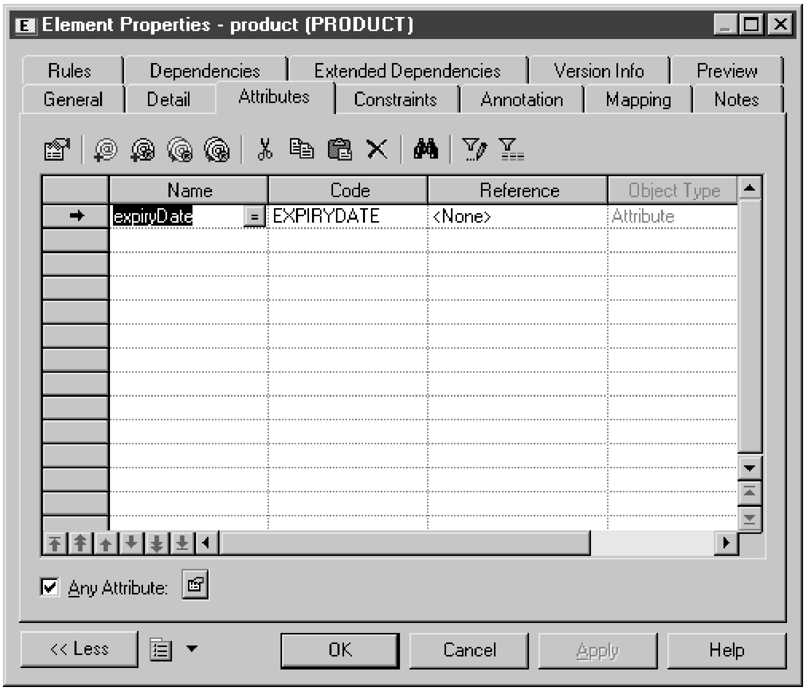
In a schema, Any Attribute is declared with the <anyAttribute> tag.

For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:attribute name="EXPIRYDATE" type="xs:date">
         </xs:attribute>
         <xs:anyAttribute namespace="##local" processContents="skip"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Caution
Any Attribute only is displayed in a schema (see the Preview tab of a model property sheet).

The Any Attribute feature is available via a check box in the bottom-left corner of an Attributes tab.



Name	Code	Reference	Object Type
expiryDate	EXPIRYDATE	<None>	Attribute

☑ Any Attribute:

To display an Any Attribute property sheet, select the Any Attribute check box and then click the Properties tool.

”Any” attribute property sheet General tab

The General tab contains the following properties:

Property	Description
ID	ID of the Any Attribute. Its value must be of type ID and unique within the model containing the Any Attribute
Names-pace	Namespaces containing the attributes that can be used. If you select **##any**, attributes from any namespace can be used. If you select **##other**, attributes from any namespace other than the target namespace of the schema can be used. If you select **##local**, attributes that are not qualified with a namespace can be used. If you select **##targetNamespace**, attributes from the target namespace of the schema can be used. If you type a white space delimited list with URI references, ##targetNamespace and ##local, attributes from this list can be used
Process contents	Indicator of how an XML processor should handle validation of XML documents containing the attributes specified by the Any Attribute. If you select **Lax**, the XML processor will try to obtain the schema and validate any attribute of the specified namespaces. If the schema cannot be found, no error will occur. If you select **Skip**, the XML processor will not try to validate the attributes of the specified namespaces. If you select **Strict**, the XML processor must obtain the schema and validate any attribute of the specified namespaces

Constraints: Keys, Uniques, and KeyRefs (XSM)

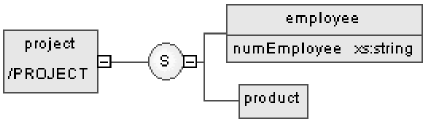
Constraints indicate that element values must be unique within their specified scope. Constraints are only available in a model targeted with XSD.

Each constraint has two specific attributes: selector and field.

In a schema, a constraint is declared with its corresponding tag: <unique>, <key> or <keyRef>.

There are three kinds of identity constraints:

- A **Unique** constraint - specifies that an element or an attribute value (or set of values) must be unique or null within a specified scope. For example:

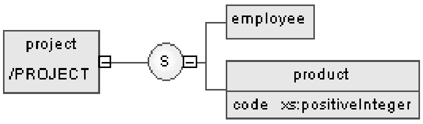


Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PROJECT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEE">
               <xs:complexType>
                  <xs:attribute name="NUMEMPLOYEE" type="xs:string">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="PRODUCT"/>
         </xs:sequence>
      </xs:complexType>
      <xs:unique name="UNIQUENUM">
         <xs:selector xpath="employee"/>
         <xs:field xpath="@numEmployee"/>
      </xs:unique>
   </xs:element>
</xs:schema>
```

The UNIQUENUM unique constraint, defined on the project element, specifies that the numEmployee attribute must be unique or null within the employee element

- A **Key** constraint - specifies that an element or an attribute value (or set of values) must be a key within a specified scope; the data must be unique, not null, and always present within a specified scope. For example:

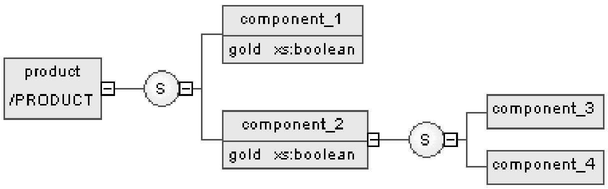


Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="PROJECT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="EMPLOYEE"/>
            <xs:element name="PRODUCT">
               <xs:complexType>
                  <xs:attribute name="CODE" type="xs:positiveInteger">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:key name="KEYCODE">
         <xs:selector xpath="product"/>
         <xs:field xpath="@code"/>
      </xs:key>
   </xs:element>
</xs:schema>
```

The KEYCODE key constraint, defined on the project element, specifies that the code attribute must be unique, not null and always present within the product element.

♦ A **KeyRef** constraint - specifies that an element or attribute value (or set of values) corresponds to the value of a specified key or unique constraint. A keyRef is a reference to a key or a unique constraint. For example:



Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:k="keyRef.namespace">
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="COMPONENT_1">
               <xs:complexType>
                  <xs:attribute name="GOLD" type="xs:boolean">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element name="COMPONENT_2">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="COMPONENT_3"/>
                     <xs:element name="COMPONENT_4"/>
                  </xs:sequence>
                  <xs:attribute name="GOLD" type="xs:boolean">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
      <xs:unique name="UNIGOLD">
         <xs:selector xpath="k:COMPONENT_1"/>
         <xs:field xpath="@GOLD"/>
      </xs:unique>
      <xs:keyref name="KEYREF_UNIGOLD" refer="k:UNIGOLD">
         <xs:selector xpath="k:COMPONENT_2"/>
         <xs:field xpath="@GOLD"/>
      </xs:keyref>
   </xs:element>
</xs:schema>
```

The KEYREF_UNIGOLD keyRef, defined on the product element, by reference to the UNIGOLD unique constraint, specifies that the gold attribute must be unique or null within the component_2 element, as well as it must be unique or null within the component_1 element (See UNIGOLD).

Creating a constraint

☞ You create constraints on the Constraints tab of an element, using one of the following tools:

Tool	Description
	Add Key Constraint - The element value must be a key within the specified scope. The scope of a key is the containing element in an instance document. A key must be unique, not null, and always present
	Add Unique Constraint - The element value must be unique or null within the specified scope
	Add KeyRef Constraint - The element value corresponds to those of the specified key or unique constraint

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Constraint properties

You can modify an object's properties from its property sheet. To open a constraint property sheet, double-click its diagram symbol or its Browser entry in the Constraints folder beneath an entity.

The General tab contains the following properties:

Property	**Description**
Name	Name of the constraint. It must be a no-colon-name (See Glossary)
Code	Code of the constraint. It must be a no-colon-name
Comment	Descriptive label of the constraint
Stereo-type	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the constraint. Its value must be of type ID and unique within the model containing the constraint.
Reference [KeyRef only]	Name of a key or a unique constraint defined in the current model (or another model with a specified namespace). The Reference value must be a qualified name (See Glossary)
Selector (XPath)	An XML Path Language expression that selects a set of elements across which the values specified in the Fields tab must be unique. There can only be one selector (see "Specifying a constraint selector" on page 57)

Constraint property sheet Fields tab

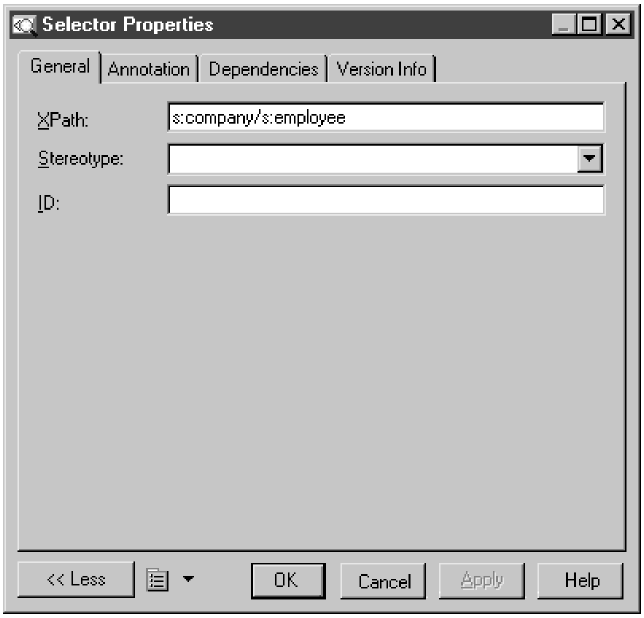
The Fields tab lists XPath expressions used to define the constraint (see "XPath abbreviated syntax" on page 59). If more than one field is listed, the combination of fields must be unique. For information, see "Specifying constraint fields" on page 58.

Specifying a constraint selector

A constraint selector specifies an XPath expression that selects a set of elements for a constraint.

❖ To specify a constraint selector

1. Open the property sheet of a constraint, and enter an XPath expression in the Selector (XPath) field (see "XPath abbreviated syntax" on page 59).
2. Click Apply to make available the Properties tool to the right of the field.
3. Click the Properties tool to open the selector property sheet.



XPath: s:company/s:employee
Stereotype:
ID:

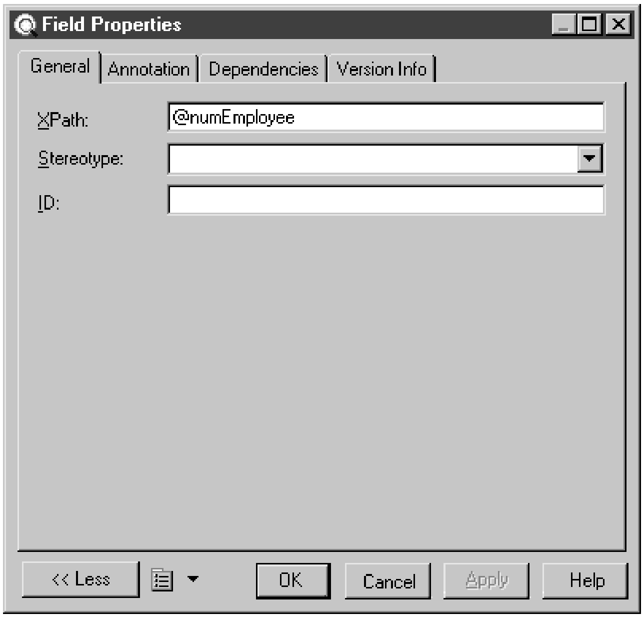
4. [optional] Enter additional properties for the selector, and then click OK to return to the constraint.

Specifying constraint fields

Constraint fields are XPath expressions used to define a constraint (unique, key or keyRef).

❖ To create an identity constraint field

1. Open the property sheet of a constraint and then click the Fields tab.
2. Click in the XPath column and enter an XPath expression (see "XPath abbreviated syntax" on page 59).
3. Click the Properties tool to open the property sheet of the field:



XPath: @numEmployee
Stereotype:
ID:

4. [optional] Enter additional properties for the selector, and then click OK to return to the constraint.

XPath abbreviated syntax

An XPath expression allows you to locate a node (an element with its ramifications) in the hierarchical tree structure of an XML document.

The XPath expressions permitted to define constraint selectors and fields are limited to a subset of the full XPath language defined in the W3C Recommendation XML Path Language 1.0:

Syntax	Description
/	Root node of the XML document. It is the root element with its ramifications
.	Selects the context node. It is the current element (on which an identity constraint is defined) with its ramifications
..	Selects the context node parent
*	Selects all the child elements of the context node
employee	Selects all the employee child elements of the context node

Syntax	**Description**
s:employee	Selects all the employee child elements of the context node, defined in the namespace with the “s” prefix
@numEmployee	Selects the numEmployee attribute of the context node
@*	Selects all the attributes of the context node
../@numEmployee	Selects the numEmployee attribute of the context node parent
employee[1]	Selects the first employee child element of the context node
employee[last()]	Selects the last employee child element of the context node
*/employee	Selects all the employee grandchildren of the context node
//employee	Selects all the employee descendants of the root node
.//employee	Selects the employee descendants of the context node
company//employee	Selects the employee descendants of the company child elements of the context node
//company/employee	Selects all the employee elements with company as parent element in the context node
/book/chapter[2]/section[3]	Selects the third section in the second chapter of the book
employee[@dept=“doc”]	Selects all the employee child elements of the context node with a dept attribute set to doc
employee[@dept=“doc”][3]	Selects the third employee child element of the context node with a dept attribute set to doc
employee[3][@dept=“doc”]	Selects the third employee child element of the context node only if it has a dept attribute set to doc
chapter[title]	Selects the chapter child elements of the context node with at least one title child element

Syntax	Description
chapter[title="About this book"]	Selects the chapter child elements of the context node with at least one title child element with a text content set to About this book
em-ployee[@numEmployee and @dept]	Selects all the employee child elements of the context node with the numEmployee and dept attributes
text()	Selects all the child nodes of the text context node

Selector and Field property sheet General tab

The General tab of a selector or field property sheet contains the following properties:

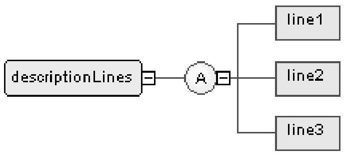
Property	Description
XPath	For a selector: An XPath expression relative to the parent element being declared. It identifies the child elements to which the identity applies For a field: An XPath expression relative to each element selected by the selector of the constraint. It identifies a single element (with a simple type) whose content or value is used for the constraint
Stereo-type	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the selector. Its value must be of type ID and unique within the model containing the selector

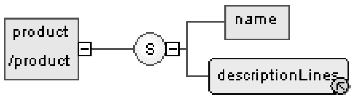
Groups (XSM)

A group of elements is a set of elements arranged by a group particle (all, choice or sequence), which is then referenced in the model by various elements.

- A **group** - is created independently, without a parent element, and can be reused multiple times by elements, complex types or other global groups, through references. In a schema, it is directly linked to the <schema> tag (root element). See "Creating a group" on page 63.
- A **reference to a group** - is created within an element, complex type or global group, and makes the referenced group available to its parent. See "Creating a reference to a group" on page 64.

For example:





The descriptionLines group is reused in the definition of the product element by clicking the sequence group particle (S) with the palette Group tool. The Reference property of the referencing group property sheet is then set to descriptionLines.

- In the generated XSD file, the group is first declared with the <group> tag and then reused through a reference (ref) set to descriptionLines:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:group name="descriptionLines">
      <xs:all>
         <xs:element name="line1"/>
         <xs:element name="line2"/>
         <xs:element name="line3"/>
      </xs:all>
   </xs:group>
   <xs:element name="product">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="name"/>
            <xs:group ref="descriptionLines">
            </xs:group>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

♦ In the generated DTD file, the group is expanded directly within its parent element:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT line1 EMPTY>
<!ELEMENT line2 EMPTY>
<!ELEMENT line3 EMPTY>
<!ELEMENT product (name,line1,line2,line3)>
<!ELEMENT name EMPTY>
```

♦ In the generated XDR file, the group is declared through a <group> tag, within an <ElementType> tag with its order attribute set to seq:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Schema name="XDR_group"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="line1" content="empty"/>
  <ElementType name="line2" content="empty"/>
  <ElementType name="line3" content="empty"/>
  <ElementType name="name" content="empty"/>
  <ElementType name="product" model="closed" content="eltOnly" order="seq">
     <element type="name"/>
     <group>
        <element type="line1"/>
        <element type="line2"/>
        <element type="line3"/>
     </group>
  </ElementType>
</Schema>
```

Groups in DTD and XDR files
In a model targeted with DTD or XDR language, there are no global or referencing groups, although they appear on the diagram. Groups are expanded within their parent element and their child elements are declared individually as global elements. (See generated DTD and XDR files in "Groups (XSM)" on page 62)

Creating a group

A group is created independently in the diagram, and will be reused within other elements by way of references (see "Creating a reference to a group" on page 64)

You can create a group in any of the following ways:

- ♦ Select the Group tool in the diagram Palette and click in an empty space in the diagram.
- ♦ Select Model ➤ Groups to access the List of Groups, and click the Add a Row tool.
- ♦ Right-click the model or package in the Browser, and select New ➤ Group.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Creating a reference to a group

A reference to a group is created as a child of an element, group or complex type, and makes the referenced group available to its parent.

You can create a referencing group in any of the following ways:

- ♦ Select the Group tool in the diagram Palette, and click on an element, group, or complex type symbol.
- ♦ On the Items tab of the property sheet of a group particle, click the Add Group with Reference to Group tool (see "Group particle property sheet Items tab" on page 42).

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Group properties

You can modify an object's properties from its property sheet. To open a group (or reference to a group) property sheet, double-click its diagram symbol or its Browser entry in the Groups folder (or, in the case of a reference to a group, beneath its parent object).

The General tab contains the following properties:

Property	Description
Name	[unavailable to references to groups] Name of the group. It must be a no-colon-name (See Glossary). Required when the group is global
Code	[unavailable to references to groups] Code of the group. It must be a no-colon-name. Required when the group is global

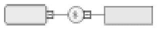
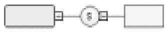
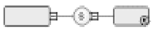
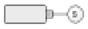
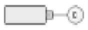
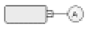
Property	Description
Comment	Descriptive label for the group
Stereo-type	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	[for references to groups only] Name of a group in the current model or another model opened in the workspace. It must be a qualified name (See Glossary). A reference allows you to reuse a group with all its properties without having to define it again. Use the list to select a group in the current model. Use the Browse tool to select a group from any model opened in the workspace. If you select a group from another model, a shortcut is created with the referencing group. When a reference is defined, the name and code properties are grayed. The name and code are those of the target group
Group type	[unavailable to references to groups] Specifies how child elements are to be used within the group. You can choose between: ♦ all ♦ choice ♦ sequence For more information, see "Group Particles (XSM)" on page 39.
Minimum	Minimum number of times the group can occur. To specify that the group is optional, set this attribute to zero.
Maximum	Maximum number of times the group can occur. For an unlimited number of times, select unbounded.

Once you have defined the reference of a referencing group, you can locate the referenced group in the diagram by right-clicking the referencing group symbol and selecting Find Referenced Group in the contextual menu. The referenced group is displayed with handles in the diagram.

You can access directly to the Preview tab of a group property sheet. Right-click a group (or a referencing group) symbol in the diagram and select Preview in the contextual menu.

Linking child objects to a group

XML objects do not support standard link objects. To link a child object to a group, you must click the child object tool in the palette and then click the group symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action
E	If you click a group symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet 
A	If you click a group symbol with the **Any** tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet 
G	If you click a group symbol with the Group tool, a sequence group particle and a referencing group are created. You can modify the group particle via its property sheet. You must now select a group for the reference 
T	If you click a group symbol with the Complex Type tool, a complex type symbol is displayed superposed, but not linked, to the group symbol. A global complex type cannot be the child of a group
S	If you click a group symbol with the Sequence tool, a sequence group particle is displayed linked to the group symbol 
C	If you click a group symbol with the Choice tool, a choice group particle is displayed linked to the group symbol 
A	If you click a group symbol with the **All** tool, an all group particle is displayed linked to the group symbol 

Pointer indications

When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Attribute Groups (XSM)

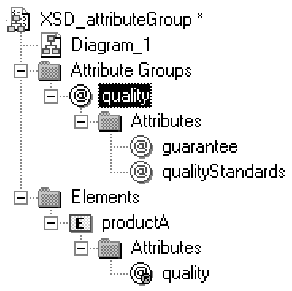
Attribute groups are not supported by XDR.

An attribute group is a set of attributes, which is referenced in the model by various elements:

♦ An **attribute group** - is created independently, without a parent element, and can be reused multiple times by elements, complex types or other global attribute groups, through references. In a schema, it is directly linked to the <schema> tag (root element). See "Creating an attribute group" on page 69.

♦ A **reference to an attribute group** - is created within an element, complex type, or global attribute group, and makes the referenced attribute group available to its parent. See "Creating a reference to an attribute group" on page 69.

For example:



The quality attribute group is composed of the guarantee and qualityStandards attributes. The productA element reuses the quality attribute group via the Attributes tab of its property sheet.

♦ Generated XSD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:attributeGroup name="quality">
      <xs:attribute name="guarantee">
      </xs:attribute>
      <xs:attribute name="qualityStandards">
      </xs:attribute>
   </xs:attributeGroup>
   <xs:element name="productA">
      <xs:complexType>
         <xs:attributeGroup ref="quality">
         </xs:attributeGroup>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

In a schema, a group of attributes is declared with the <attributeGroup> tag. It can contain the following tags: <attribute>, <attributeGroup> or <anyAttribute>.

- Generated DTD file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT productA EMPTY>
<!ATTLIST productA
          guarantee            CDATA
          qualitystandards     CDATA>
```

Creating an attribute group

An attribute group is created independently, and will be reused within other elements by way of references (see “Creating a reference to an attribute group” on page 69)

You can create an attribute group in any of the following ways:

- Select Model ➤ Attribute Groups to access the List of Attribute Groups, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select New ➤ Attribute Group.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Creating a reference to an attribute group

A reference to an attribute group is created within an element, complex type, or attribute group, and makes the referenced attribute group available to its parent. You can create a reference to an attribute group as follows:

- On the Attributes tab of the property sheet of an element, complex type, or attribute group, click the Add Group with Reference to Group tool (see “Element property sheet Attributes tab” on page 35).

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Attribute group properties

You can modify an object’s properties from its property sheet. To open an attribute group (or reference to an attribute group) property sheet, double-click its Browser entry in the Attribute Groups folder (or, in the case of a reference to an attribute group, beneath its parent object). The following sections detail the property sheet tabs that contain the properties most commonly entered for attribute groups.

The General tab contains the following properties:

Property	Description
Name	[unavailable to references to attribute groups] Name of the attribute group. It must be a no-colon-name (See Glossary). Required when the attribute group is global
Code	[unavailable to references to attribute groups] Code of the attribute group. It must be a no-colon-name. Required when the attribute group is global
Comment	Descriptive label of the attribute group
Stereo-type	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Reference	[for references to attribute groups only] Name of an attribute group from the current model or any model opened in the workspace. It must be a qualified name (See Glossary). If you select an attribute group from another model, a shortcut is created with the referencing attribute group. A reference allows you to reuse an attribute group with all its properties without having to define it again. When a reference is defined, the name and code properties are grayed. The name and code are those of the target attribute group
ID	ID of the attribute group. Its value must be of type ID and unique within the model containing this attribute group

Attribute group property sheet Attributes tab

The Attributes tab lists the attributes and attribute groups associated with the attribute group.

For information about the tools available on this tab for adding attributes and attribute groups, see "Creating an attribute" on page 47.

Simple Types (XSM)

You can only create simple types in a model targeted with XSD.

A simple type is a data type definition for elements or attributes with text-only content. It cannot contain elements or attributes.

A simple type is defined by derivation of an existing simple type (built-in data type or derived simple type). There are three kinds of derivation for a simple type:

- List - contains a white space-separated list of values of an inherited simple type
- Restriction - has a range of values restricted to a subset of those of an inherited simple type
- Union - contains a union of values of two or more inherited simple types

☞ For more information on simple type derivations, see section "Derivations: Extensions, Restrictions, Lists and Unions (XSM)" on page 79.

Once defined in a model, a simple type can be reused in the definition of an attribute, an element or a complex type.

Example of a simple type in a schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="BARCODE">
     <xs:restriction base="xs:nonNegativeInteger" id="STR1">
        <xs:length value="13"/>
     </xs:restriction>
   </xs:simpleType>
   <xs:element name="PRODUCTA" type="BARCODE"/>
   <xs:element name="PRODUCTB" type="BARCODE"/>
</xs:schema>
```

Creating a simple type

You can create a simple type in any of the following ways:

- Select Model ➤ Simple Types to access the List of Simple Types, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select New ➤ Simple Type.

Caution
If the simple type symbol does not appear in the diagram, select Show Symbols in the Symbol menu, then click the Simple Type tab and select the simple type box to display its symbol in the diagram.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Simple type properties

You can modify an object's properties from its property sheet. To open a simple type property sheet, double-click its diagram symbol or its Browser entry in the Simple Types folder.

The General tab contains the following properties:

Property	Description
Name	Name of the simple type. It must be a no-colon-name (See Glossary). If specified, it must be unique among all simple types and complex types
Code	Code of the simple type. It must be a no-colon-name. If specified, it must be unique among all simple types and complex types
Comment	Descriptive label of the simple type
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Derivation	Derivation method for the simple type. Enabled and required when the simple type is defined
Final	Property to prevent derivation of the current simple type
ID	ID of the simple type. Its value must be of type ID and unique within the model containing the simple type

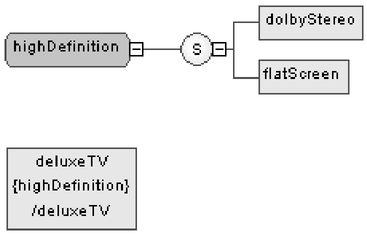
Complex Types (XSM)

You can only create complex types in a model targeted with XSD.

A complex type is a data type definition used to define attributes and child elements of a parent element. It is a template for a data type definition that can be reused and derived by extension or restriction.

A complex type has:

- a global scope when it has no parent element in the diagram and when it is directly linked to the <schema> tag. It can then be reused or derived, by extension or restriction, in other parts of the schema.
- a local scope when integrated into an <element> tag. It applies only to its containing element.



In the example above, HighDefinition is a global complex type, reused as data type for the deluxeTV element.

Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:complexType name="highDefinition">
      <xs:sequence>
         <xs:element name="dolbyStereo"/>
         <xs:element name="flatScreen"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="deluxeTV" type="highDefinition"/>
</xs:schema>
```

Caution
Global complex types appear in the model as objects, with their corresponding symbol in the diagram. Local complex types only appear in the schema (see Preview tab of an element property sheet).

Creating a complex type

You can create a complex type in any of the following ways:

- Use the Complex Type tool in the diagram Palette.
- Select Model ➤ Complex Types to access the List of Complex Types, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select New ➤ Complex Type.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Complex type properties

You can modify an object's properties from its property sheet. To open a complex type property sheet, double-click its diagram symbol or its Browser entry in the Complex Types folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for complex types.

The General tab contains the following properties:

Property	Description
Name	Name of the complex type. It must be a no-colon-name (See Glossary) and unique among all simple types and complex types
Code	Code of the complex type. It must be a no-colon-name and unique among all simple types and complex types
Comment	Descriptive label of the complex type
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Group type	Specifies that the complex type has child elements, and how they are used. You can choose between: ♦ all – All children may be present. See “Group Particles (XSM)” on page 39 ♦ choice – Only one child must be present. See “Group Particles (XSM)” on page 39 ♦ group – Reference to a predefined group. See “Groups (XSM)” on page 62 ♦ sequence – All children must be present in order. See “Group Particles (XSM)” on page 39

Property	**Description**
Content	Content type of the complex type. You can choose between: ♦ Simple - the complex type cannot contain child elements ♦ Complex - the complex type can contain child elements For more information, see "Specifying the type of content of a complex type" on page 78.
Derivation	Derivation method for the complex type. Once you have selected a derivation method, you must define a base type. Click the Properties tool beside the derivation box to display the derivation property sheet. In the General tab, select a base type in the Base Type list

Complex type property sheet Detail tab

The Detail tab contains the following properties:

Property	**Description**
Final	Property to prevent derivation of the current complex type
Block	Property to prevent another complex type with the specified type of derivation from being used in place of the current complex type
Mixed	If selected, this property indicates that character data is allowed to appear between the child elements of the current complex type. Select Mixed only if the current complex type has a complex content (See general properties)
Abstract	If selected, this property indicates that the complex type can be used in the instance document
ID	ID of the complex type. Its value must be of type ID and unique within the model containing this complex type

Complex type property sheet Attributes tab

Attributes (see "Attributes (XSM)" on page 46) give additional information about a complex type. The Attributes tab lists the attributes and attribute groups associated with the complex type.

For information about the tools available on this tab for adding attributes, see "Creating an attribute" on page 47.

Complex type property sheet Mappings tab

Object mapping is the ability to establish a correspondence between objects belonging to heterogeneous models and diagrams.

The Mapping tab of a complex type property sheet allows you to map the current complex type to PDM or OOM objects.

Select a data source in the Mapping for list. If it is the first time you define a mapping for a complex type, the Mapping for list is empty. Click the Add a Mapping for a Data Source tool and select a data source.

Complex Type Sources tab

The Complex Type Sources tab allows you to associate one or several abstract data types (in PDMs) or classes (in OOMs) to the current complex type.

You can use the Add Objects tool to select abstract data types or classes from the PDMs or OOMs opened in the current workspace.

Attributes Mapping tab

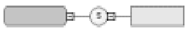
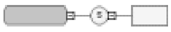
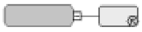
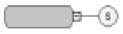
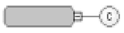
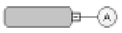
The Attributes Mapping tab allows you to define the mapping between abstract data type attributes (in PDMs) or class attributes (in OOMs) and attributes in the current complex type.

Icon	Description
▦	**Add Mapping** - To select the attributes in the current complex type that will be mapped to abstract data type attributes or class attributes. Once you have selected the attributes, you can use the list in the Mapped to column to select corresponding abstract data type attributes or class attributes
▦	**Create from Sources** - To copy abstract data type attributes or class attributes to the current complex type attributes
▦	**Generate Mapping** - To automatically generate a mapping between abstract data type attributes or class attributes and complex type attributes with same name or code in the data source and the current model

☞ For more information on complex type mapping, see section Mapping objects in an XML model in chapter Working with an XML model.

How to link a child object to a complex type?

XML objects do not support standard link objects. To link a child object to a complex type, you must click the child object tool in the palette and then click the complex type symbol in the diagram. This will automatically create a link between both objects. See the following table for allowed links:

Tool	Action
→E	If you click a complex type symbol with the Element tool, a sequence group particle and a child element symbol are created. You can modify the group particle via its property sheet 
A	If you click a complex type symbol with the **Any** tool, a sequence group particle and an any symbol are created. You can modify the group particle via its property sheet 
G	If you click a complex type symbol with the Group tool, a referencing group is created. You can modify the group particle via its property sheet. You must now select a group for the reference 
⊘T	If you click a complex type symbol with the Complex Type tool, a second complex type symbol is displayed superposed, but not linked, to the first complex type symbol. A complex type cannot be the child of another complex type
S	If you click a complex type symbol with the Sequence tool, a sequence group particle is displayed linked to the complex type symbol 
C	If you click a complex type symbol with the Choice tool, a choice group particle is displayed linked to the complex type symbol 
A	If you click a complex type symbol with the **All** tool, an all group particle is displayed linked to the complex type symbol 

Pointer indications
When you cannot click a symbol or an empty space in a diagram, the Pointer displays a forbidden sign (See complex type in Tool column). When there is a possibility to create a symbol above, below or next to another one, the Pointer displays an arrow indicating the corresponding direction (See element in Tool column).

Specifying the type of content of a complex type

A complex type can contain either:

- simple content – character data or a simple type (but no elements), or
- complex content – elements or elements and character data

❖ To specify the type of content of a complex type

1. Open the property sheet of a complex type and select either simple or complex in the Content list.
2. Click Apply to make available the Properties tool to the right of the list.
3. [optional] Click the Properties tool to open the property sheet of the content, and specify an ID and, in the case of complex content, whether the content can be mixed.
4. Click OK to return to the diagram.

Content property sheet General tab

The General tab contains the following properties:

Property	Description
ID	ID for the complex content. Its value must be of type ID and unique within the model containing the complex content
Mixed [complex content only]	Specifies that character data is allowed to appear between child elements of the complex type.

Derivations: Extensions, Restrictions, Lists and Unions (XSM)

You can use derivations to extend or restrict the values of elements and simple and complex types.

An XML model allows you to derive:

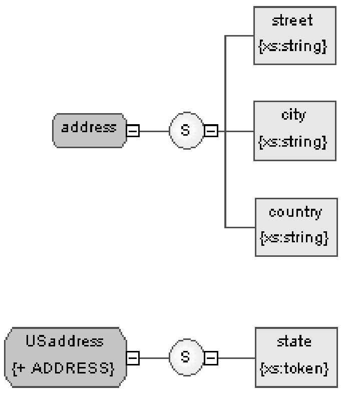
- ♦ Elements by extension, restriction, list or union
- ♦ Simple types by restriction, list or union
- ♦ Complex types by extension or restriction

Derivation in element property sheet
When you define a derivation in an element property sheet, a simple or a complex type is automatically created within the element declaration (See Preview tab). The Embedded type property is automatically set to Simple or Complex, and the Content property to Simple or Complex in the case of an embedded complex type.

Deriving by extension

You can derive an element or complex type by extension to extend the values of its base type.

For example:



USaddress is a derivation by extension of the address complex type.

Generated schema:

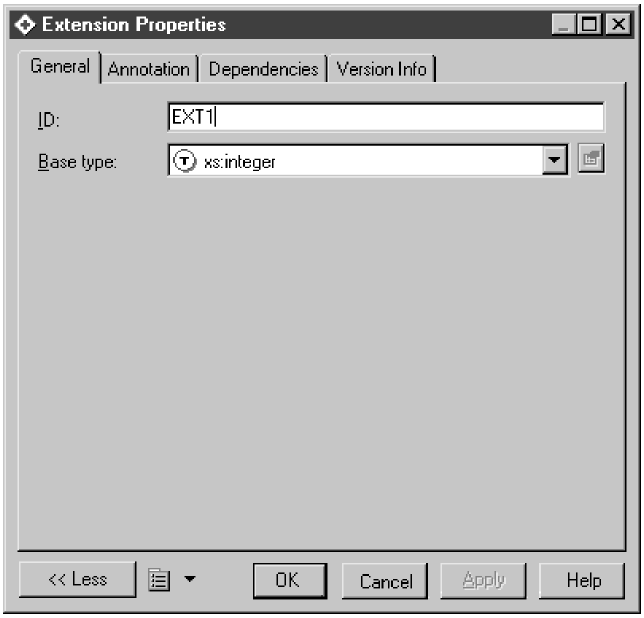
```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:complexType name="ADDRESS">
      <xs:sequence>
         <xs:element name="STREET" type="xs:string"/>
         <xs:element name="CITY" type="xs:string"/>
         <xs:element name="COUNTRY" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="USADDRESS">
      <xs:complexContent>
         <xs:extension base="ADDRESS">
           <xs:sequence>
              <xs:element name="STATE" type="xs:token"/>
           </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

❖ To define a derivation by extension

1. Open the property sheet of an element or complex type and select Extension in the Derivation list.

   The Content field (and, in the case of an element, the Embedded type field) is set to Complex.

2. Click the Properties tool to the right of the Derivation box to open the property sheet of the extension:



Extension Properties

General | Annotation | Dependencies | Version Info

ID: EXT1

Base type: xs:integer

<< Less | OK | Cancel | Apply | Help

3. Specify an ID, select a base type, and then click OK to return to the

element or complex type.

Extension property sheet General tab

The General tab contains the following properties:

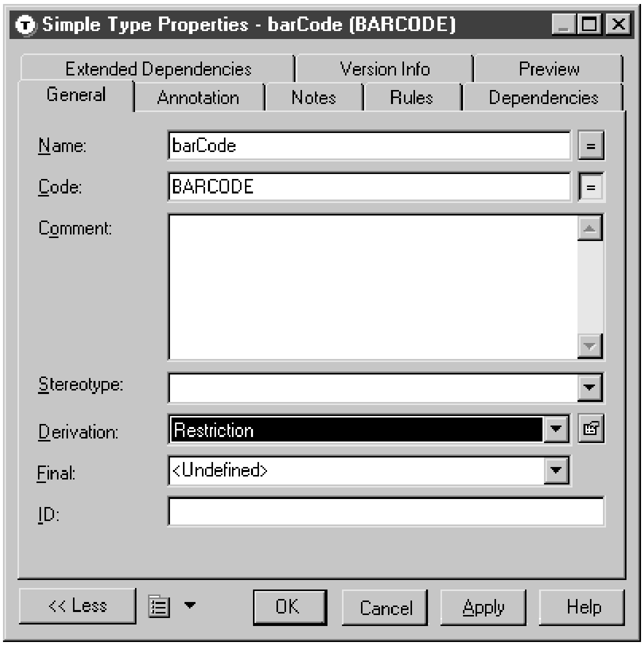
Property	Description
ID	ID of the extension. Its value must be of type ID and unique within the model containing the extension
Base Type	Data type on which the extension is based

Deriving by restriction

You can derive an element, simple type, or complex type by restriction to restrict the values of their base type.

❖ To create a restriction on an element, simple type, or complex type

1. Open the property sheet of an element, simple type, or complex type, and select Restriction in the Derivation list.



Name: barCode
Code: BARCODE
Comment:
Stereotype:
Derivation: Restriction
Final: <Undefined>
ID:

For elements and complex types, the Content field (and, in the case of an element, the Embedded type field) is set to Complex.

2. Click the Properties tool to the right of the Derivation box to open the restriction property sheet.

3. Type an ID and a base type for the restriction.
4. [simple types only] If you select Embedded type, the base type disappears and a simple type is created in the schema within the current simple type. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a base type for the embedded simple type.
5. [option – simple types only] Click the Detail tab and enter appropriate facets for the restriction (see "Restriction property sheet Detail tab" on page 82).
6. [option – simple types only] Click the Enumerations tab and enter appropriate enumerations for the restriction (see "Restriction property sheet Enumerations tab" on page 84).
7. [option – simple types only] Click the Patterns tab and enter appropriate patterns for the restriction (see "Restriction property sheet Patterns tab" on page 85).
8. Click OK to close the restriction property sheet and return to the element, simple type, or complex type.

Restriction property sheet General tab

The General tab contains the following properties:

Property	Description
ID	ID of the simple type restriction. Its value must be of type ID and unique within the model containing the simple type restriction
Base type	Data type on which the restriction is based. Select a data type in the Base type list or with the Browse tool
Embedded type [simple types only]	If selected, the base type disappears and a simple type is created in the schema within the current simple type

Restriction property sheet Detail tab

Facets are the constraints on the set of values of a simple type. The Details tab lists the facets associated with the restriction:

Icon	Facet
ⓛ	**Length** - Exact number of characters or list items allowed. It must be equal to or greater than zero

Icon	Facet
L<	**Minimum length** - Minimum number of characters or list items allowed. It must be equal to or greater than zero
L>	**Maximum length** - Maximum number of characters or list items allowed. It must be equal to or greater than zero
<	**Minimum exclusive** - Lower bound for numeric values. All values are greater than this value
>	**Maximum exclusive** - Upper bound for numeric values. All values are lower than this value
<=	**Minimum inclusive** - Minimum value allowed for data type
>=	**Maximum inclusive** - Maximum value allowed for data type
T	**Total digits** - Exact number of decimal digits allowed. It must be greater than zero
F	**Fraction digits** - Maximum number of decimal digits in the fractional part
W	**Whitespace** - Way of handling white spaces. If the value is **Preserve**, white spaces are unchanged. If the value is **Replace**, tabs, line feeds and carriage returns are replaced with spaces. If the value is **Collapse**, contiguous sequences of spaces are collapsed to a single space. Leading and trailing spaces are removed

A facet icon is displayed in the title bar of a facet property sheet.

Enumeration and Pattern facets
For Enumeration and Pattern facets, click their corresponding tabs in the restriction property sheet and double-click the arrow left of an enumeration or a pattern value to display its property sheet.

Caution
Facets only appear in the schema, within a simple type declaration (see Preview tab in the model property sheet).

Facet general properties

The General tab of a facet property sheet displays the following properties:

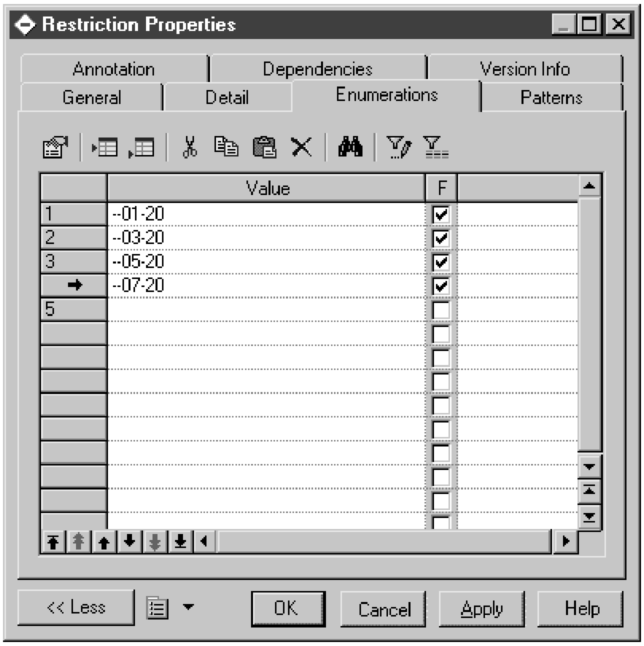
Property	Description
ID	ID of the facet. Its value must be of type ID and unique within the model containing the facet
Value	Value(s) of the facet
Fixed	To prevent a modification of the facet value(s), select the Fixed property

Restriction property sheet Enumerations tab

The Enumerations tab is only available for simple type restrictions. It allows you to enter a set of acceptable values for the simple type restriction.

Select F (for Fixed) at the end of a row if you want to prevent the modification of a value.

For example: the meetings simple type, based on the xs:gMonthDay data type, is restricted to the following dates: 01/20, 03/20, 05/20 and 07/20.



	Value	F
1	--01-20	✓
2	--03-20	✓
3	--05-20	✓
➜	--07-20	✓
5		

Generated schema:

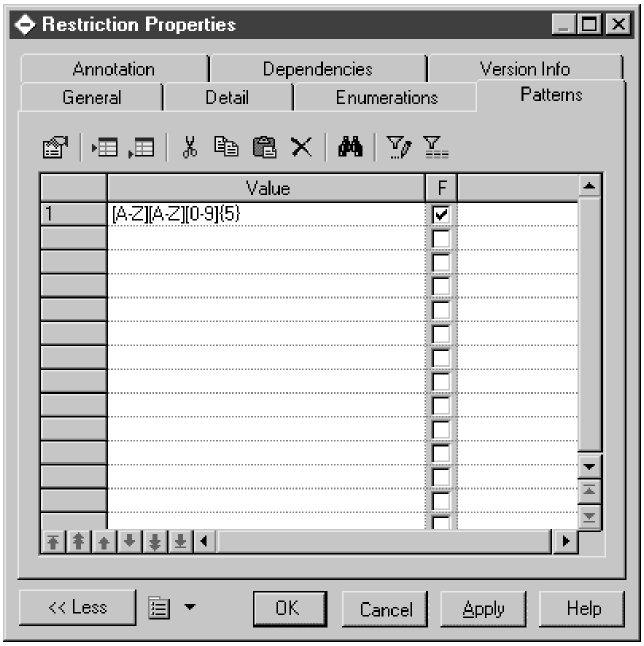
```
<xs:simpleType name="MEETINGS">
  <xs:restriction base="xs:gMonthDay">
     <xs:enumeration value="--01-20"/>
     <xs:enumeration value="--03-20"/>
     <xs:enumeration value="--05-20"/>
     <xs:enumeration value="--07-20"/>
     <xs:enumeration/>
  </xs:restriction>
</xs:simpleType>
```

Restriction property sheet Patterns tab

The Patterns tab is only available for simple type restrictions. It allows you to enter the exact sequence of acceptable values for the simple type restriction.

Select F (for Fixed) at the end of a row if you want to prevent the modification of a value.

For example: the zipCode simple type, based on the xs:string data type, is restricted to the following pattern: two uppercase letters, from A to Z, followed by a five-digit number, each digit ranging from 0 to 9.



	Value	F
1	[A-Z][A-Z][0-9]{5}	✓

Generated schema:

```
<xs:simpleType name="ZIPCODE">
  <xs:restriction base="xs:string">
     <xs:pattern value="[A-Z][A-Z][0-9]{5}"/>
  </xs:restriction>
</xs:simpleType>
```

Deriving by list

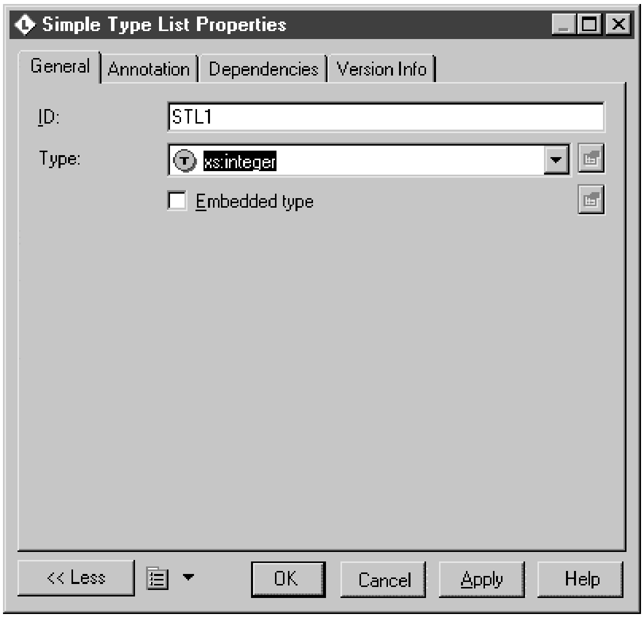
You can derive an element or simple type by list to define it as a list of values of a specified data type.

❖ To define an element or simple type derivation by list

1. Open the property sheet of an element or simple type, and select List in the Derivation list.

   For elements, the Embedded type field is set to Simple.

2. Click the Properties tool to the right of the Derivation box to open the list property sheet:



ID: STL1

Type: xs:integer

☐ Embedded type

3. Enter an ID and a type for the list.

   If you select Embedded type, the type disappears and a simple type is created in the schema within the current simple type. Click Apply, and then the Properties tool beside the Embedded type box, to define a derivation and a type for the embedded simple type.

4. Click OK to close the list property sheet and return to the element or simple type.

List property sheet General tab

The General tab contains the following properties:

Property	**Description**
ID	ID of the simple type list. Its value must be of type ID and unique within the model.
Type	Data type for the list of values
Embed-ded Type	If selected, the type disappears and a simple type is created in the schema within the current simple type or element

Deriving by union

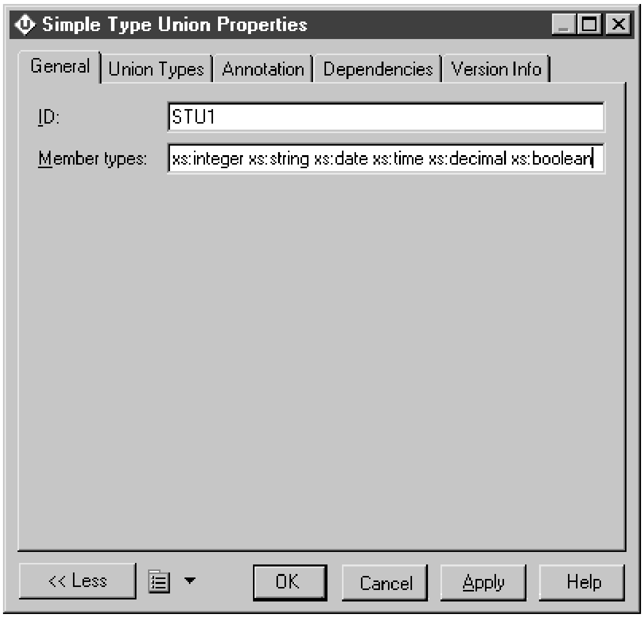
You can derive an element or simple type by union to define it as a collection of built-in and simple data types.

❖ To define an element or simple type derivation by union

1. Open the property sheet of an element or simple type, and select Union in the Derivation list.

   For elements, the Embedded type field is set to Simple.

2. Click the Properties tool to the right of the Derivation box to open the union property sheet:



ID: STU1
Member types: xs:integer xs:string xs:date xs:time xs:decimal xs:boolean

3. Enter an ID and member types for the union.

4. [optional] Click the Union Types tab and add appropriate simple types to the union.

5. Click OK to close the union property sheet and return to the element or simple type.

Union property sheet General tab

The General tab of a simple type union property sheet displays the following properties:

Property	**Description**
ID	ID of the simple type union. Its value must be of type ID and unique within the model containing the simple type union
Member Types	White space separated list of built-in data types. Values must be qualified names (See Glossary)

Union property sheet Union Types tab

The Union Types tab of a simple type union property sheet displays a list where you can add simple types using the Add Simple Type tool.

Annotations (XSM)

Annotations are only available in models targeted with XSD.

You define annotations when you want to add information about an XML model. There are three kinds of annotations:

- Annotation - provides extra information on XML models or schemas. You can define multiple annotations at this level, and each can contain multiple documentation and/or application information tags.
- Documentation – is contained within an annotation and contains an URI reference or any well-formed XML content that gives extra information about XML objects or documents
- Application Information - is contained within an annotation and contains an URI reference or any well-formed XML content that is used by applications for processing instructions

Generated schema of a global annotation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:annotation id="ANNOT1">
      <xs:documentation source="attributes.dtd" xml:lang="en-US">
        <?xml version="1.0" encoding="UTF-8" ?>
        <!ELEMENT COMPANY (EMPLOYEES,PRODUCTS,CLIENTS)>
        <!--  -->
        <!ATTLIST COMPANY
           FOUNDATION                         CDATA
           STATUS                             CDATA>
        <!ELEMENT EMPLOYEES EMPTY>
        <!--  -->
        <!ATTLIST EMPLOYEES
           NUMBER                             CDATA>
        <!ELEMENT PRODUCTS EMPTY>
        <!--  -->
        <!ATTLIST PRODUCTS
           REFERENCES                         CDATA
           QUANTITIES                         CDATA>
        <!ELEMENT CLIENTS EMPTY>
        <!--  -->
        <!ATTLIST CLIENTS
           FIDELITY                           CDATA
           SOLVENCY                           CDATA>
        </xs:documentation>
      <xs:appinfo source="http://www.parsersandco.com"></xs:appinfo>
   </xs:annotation>
</xs:schema>
```

This global annotation is composed of a documentation, with a well-formed XML content (extract of a DTD file), and an application information.

Creating an annotation

You can create a annotation at the global level in any of the following ways:

- ♦ Open the Items or External Schemas tab in the property sheet of the model, and click the Add Annotation tool.
- ♦ Right-click the model or package in the Browser, and select New ➤ Annotation.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Creating a documentation or application information

You can create a documentation or application information using the following tools on the:

- ♦ Items tab of an annotation property sheet - to add content to an annotation at the global, schema level
- ♦ Annotations tab of an element or other object property sheet - to add content to an annotation at the global, schema level

Tool	Description
	Add Documentation - Adds a comment or a document reference to be read by users
	Add Application Information - Adds an information to be used by applications for processing instructions

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Annotation property sheet General tab

The General tab contains the following properties:

Property	Description
ID	Must be of type ID and unique within the model containing the annotation.

Annotation property sheet Items tab

The Items tab lists the documentation and application information tags contained within the annotation. For information on the tools for creating content available on this tab, see "Creating a documentation or application information" on page 90:

Documentation and application information property sheet General tab

The General tab contains the following properties:

Property	Description
Source	Source of the content. It must be a URI reference
Language	[documentation only] Language used in the documentation. For example: en, en-GB, en-US, de, fr

Documentation and application information property sheet Content tab

The Content tab allows you to write or paste any well-formed XML content.

Notations (XSM)

Notations allow you to define and process non-XML objects within an XML model.

For example: picture files with a .GIF extension.

Generated schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!--
   Integrating GIF files in your XML model
   -->
   <xs:notation name="PICTURES" public="pictures/gif"
 system="user/local/pictureViewer"/>
</xs:schema>
```

Notations are not available on models targeted with XDR.

Creating a notation

You can create a notation in any of the following ways:

- ♦ Select Model ➤ Notations to access the List of Notations, and click the Add a Row tool.
- ♦ Right-click the model or package in the Browser, and select New ➤ Notation.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Notation properties

You can modify an object's properties from its property sheet. To open a notation property sheet, double-click its diagram symbol or its Browser entry in the Notations folder.

The General tab contains the following properties:

Property	Description
Name	Name of the notation. It must be a no-colon-name (See Glossary)
Code	Code of the notation. It must be a no-colon-name
Comment	Descriptive label of the notation
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
ID	ID of the notation. Its value must be of type ID and unique within the model containing the notation
Public	URI reference identifying the non-XML object. For example: pictures/gif
System	URI reference identifying the application that will process the non-XML object. For example: user/local/pictureViewer

Entities (XSM)

Entities enable you to include predefined values, external XML or non-XML files in an XML model targeted with a DTD.

When an XML processor reads an entity reference in an XML document, it will replace this entity reference by its value defined in the DTD file of the XML document.

An entity reference is the entity name preceded by an ampersand and followed by a semicolon.

For example: &glossary; will be replaced by See Glossary.

The W3C has predefined five entities for XML tags:

Entity name	Reference	Value
Less than	&lt;	<
Greater than	&gt;	>
Ampersand	&amp;	&
Apostrophe	&apos;	‘
Quotation	&quot;	”

In an XML model, you just need to type the name and the value of an entity.

Creating an entity

You can create an entity in any of the following ways:

- Select Model ➤ Entities to access the List of Entities, and click the Add a Row tool.
- Right-click the model or package in the Browser, and select New ➤ Entity.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Entity properties

You can modify an object’s properties from its property sheet. To open an entity property sheet, double-click its diagram symbol or its Browser entry in the Entities folder.

The General tab contains the following properties:

Property	Description
Name	Name of the entity. It must be a no-colon-name (See Glossary)
Code	Code of the entity. It must be a no-colon-name
Comment	Descriptive label of the entity
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined
Value	Value of the entity. A string of characters in the case of a predefined value. A URI in the case of an XML or a non-XML file. For example: http://something.com/pictures/logo.gif
Public	URI reference identifying the non-XML object. For example: pictures/gif
System	URI reference identifying the application that will process the non-XML object. For example: user/local/pictureViewer
Notation	Used to define and process non-XML objects within an XML model
Parameter	If selected, the entity is parsed within the DTD, and not within the XML document as for a general entity. A parameter entity allows you to predefine a value within a DTD. This predefined value can then be easily changed within the DTD

Instructions: Import, Include and Redefine (XSM)

Import, Include and Redefine allow you to enrich your XML model with external namespaces, schema files or schema components.

These instructions are only available in a model targeted with XSD.

Imports

An import identifies a namespace whose schema components are referenced by the current schema, allowing you to use components from any schema with different target namespace than the current schema.

In a schema, an import is declared with the <import> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:import id="IMP1" namespace="xml.ordering"
 schemaLocation="ORDER.xsd"/>
</xs:schema>
```

Includes

An include allows you to include a specified schema file in the target namespace of the current schema, allowing you to use components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, an include is declared with the <include> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:include id="INC1" schemaLocation="PROFORMA.xsd"/>
</xs:schema>
```

Redefines

A redefine allows you to redefine simple and complex types, groups and attribute groups from an external schema file in the current schema, allowing you to use components from any schema with the same target namespace as the current schema or with no specified target namespace.

In a schema, a redefine is declared with the <redefine> tag. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:redefine id="REDEF1" schemaLocation="CUSTOMERS.xsd">
      <xs:group name="PRIVILEGED_CUSTOMERS">
         <xs:all>
            <xs:element name="CUSTOMER_1"/>
            <xs:element name="CUSTOMER_2"/>
            <xs:element name="CUSTOMER_3"/>
         </xs:all>
      </xs:group>
      <xs:complexType name="PRIVILEGE">
      </xs:complexType>
   </xs:redefine>
</xs:schema>
```

Creating an import, include, or redefine instruction

You can create an import, include, or redefine instruction in any of the following ways:

- Select Model ➤ Import, Include, or Redefine to access the relevant list, and click the Add a Row tool.
- Open the External Schemas tab in the property sheet of the model, and click the Add Import, Add Include, or Add Redefine tool.
- Right-click the model or package in the Browser, and select New ➤ Import, Include, or Redefine.

☞ For general information about creating objects, see the Objects chapter in the *Core Features Guide* .

Import, include, and redefine properties

You can modify an import, include, or redefine instruction's properties from its property sheet. To open an instruction property sheet, double-click its diagram symbol or its Browser entry in the Imports, Includes, or Redefines folder. The following sections detail the property sheet tabs that contain the properties most commonly entered for instructions.

The General tab contains the following properties:

Property	**Description**
Schema location	URI reference for the location of a schema file with an external namespace. You can use the Browse tool beside the Properties tool to select a schema file among those opened in the current workspace. For example: ORDER.xsd.
ID	ID of the instruction. Its value must be of type ID and unique within the schema containing the instruction.
Names-pace	[import only] URI reference for the namespace to import. For example: xml.ordering.
Comment	Descriptive label of the instruction.
Stereotype	Sub-classification used to extend the semantics of an object without changing its structure. It can be predefined or user-defined.

Redefine property sheet Items tab

The Items tab lists the items to be redefined:

Tool	Description
	Add Group - Adds a group of elements to be redefined.
	Add Attribute Group - Adds a group of attributes to be redefined.
	Add Simple Type - Adds a simple type to be redefined.
	Add Complex Type - Adds a complex type to be redefined.

Business Rules (XSM)

A business rule is a rule that your business follows. It is a written statement specifying what an information system must do or how it must be structured. It could be a government-imposed law, a customer requirement, or an internal guideline.

You can attach business rules to your model objects to guide and document the creation of your model. For example, the rule “an employee belongs to only one division” can help you graphically build the link between an employee and a division.

☞ For more information, see “Business Rules” section in the Objects chapter of the *Core Features Guide* .

CHAPTER 3

Working with an XML model

About this chapter

This chapter describes a number of basic features available with XML models.

Contents

Generating Other Models from an XML Model

You can generate another XSM from your XSM. When changes are made to the source model, they can then be easily propagated to the generated models using the Update Existing Model generation mode.

❖ To generate one XSM from another

1. Select Tools ➤ Generate XML Model (Ctrl+Shift+M) to open the XML Model Generation Options Window:
2. On the General tab, select a radio button to generate a new or update an existing model, and complete the appropriate options.
3. [optional] Click the Detail tab and set any appropriate options. We recommend that you select the Check model checkbox to check the model for errors and warnings before generation.
4. [optional] Click the Target Models tab and specify the target models for any generated shortcuts.
5. [optional] Click the Selection tab and select or deselect objects to generate.
6. Click OK to begin generation.

Generation options
For detailed information about the options available on the various tabs of the Generation window, see the Linking and Synchronizing Models chapter of the *Core Features Guide* .

Checking an XML Model

The XML Model is a very flexible tool, which allows you quickly to develop your model without constraints. You can, however, check the validity of your XSM at any time.

Check your XSM before generating an XML document
We recommend that you check your XSM before generating an XML document or other model from it. If the check encounters errors, generation will be stopped. The Check model option is enabled by default in the Generation dialog box.

You can check an XSM in any of the following ways:

- Press F4, or
- Select Tools ➤ Check Model, or
- Right-click the diagram background and select Check Model from the contextual menu

The Check Model Parameters window opens, which allows you to specify the kinds of checks to perform, and the objects to apply them to. For detailed information about this window and correcting problems reported, see “Checking a Model” section in the Models chapter of the *Core Features Guide* .

The following sections document the XSM-specific checks available by default. For information about other checks available by default for all model types, see “Checking a Model” in the Models chapter of the *Core Features Guide* .

Group particle checks

The following XML model checks are made on group articles:

Check	Description and Correction
Existence of particle	A group particle must contain elements, groups, group particles and/or Any. Manual correction: Add items to the group particle or delete it Automatic correction: None
Invalid cardinality	You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. This check is only available in a model targeted with XDR. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Model checks

This check only applies to models built on a schema.

The following XML model checks are made on models:

Check	Description and Correction
Identifier uniqueness	Two or more objects cannot have the same identifier (ID). Manual correction: Give a unique identifier to each object Automatic correction: None
Undefined identifier	You must define an identifier (ID) for each object in the model. Manual correction: Define an identifier for each object Automatic correction: None
Shortcut code uniqueness	Two shortcuts with the same code cannot be in the same namespace. Manual correction: Change the code of one of the shortcuts Automatic correction: None

Check	Description and Correction
Undefined target namespace	You should define a target namespace to your model. Manual correction: Type a URI for the Target Namespace property in the Detail tab of the model property sheet Automatic correction: None
Missing namespaces	There should be at least one namespace defined for the model. Manual correction: Type a URI and a prefix in the Namespaces tab of the model property sheet Automatic correction: Adds the target namespace URI and a prefix “ns” followed by a number (e.g. “ns1”)

Data source checks

The following XML model checks are made on data sources:

Check	Description and Correction
Data source name and code uniqueness	Data sources names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of model	A data source must have at least one model in its definition. Manual correction: Add a model from the Models tab of the data source property sheet Automatic correction: Deletes data source without a model

Check	Description and Correction
Data source containing models with different Object Language or DBMS types	The models in a data source represent a single set of information. This is why the models in the data source should share the same DBMS or object language. Manual correction: Delete models with different DBMS or object language, or modify the DBMS or object language of models in the data source Automatic correction: None

Entity checks

The following XML model checks are made on entities:

Check	Description and Correction
Entity name and code uniqueness	Entity names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined entity	You must define an entity. In the entity property sheet, you must either type a value (string of characters or URI) in the Value box, or a URI in the Public or System boxes. Manual correction: Type a value in the Value box or a URI in the Public or System boxes Automatic correction: None

Include checks

The following XML model checks are made on includes:

Check	Description and Correction
Undefined schema location	You must define a schema location for an include. Manual correction: Define a URI or select a schema file for the schema location. For example: proforma.-xsd Automatic correction: None

Simple type checks

The following XML model checks are made on simple types:

Check	Description and Correction
Simple type name and code uniqueness	Simple type names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code

Complex type checks

The following XML model checks are made on complex types:

Check	Description and Correction
Complex type name and code uniqueness	Complex type names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of attribute	A complex type should have at least one attribute. Manual correction: Define an attribute for the complex type Automatic correction: None
Existence of particle	A complex type must contain elements, groups, group particles and/or Any. Manual correction: Add items to the complex type or delete complex type Automatic correction: None

Element checks

The following XML model checks are made on elements:

Check	Description and Correction
Element name and code uniqueness	Element names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined type	An element without a reference should have a defined data type. Manual correction: In the element property sheet, define a data type with the Type list or the Browse tool Automatic correction: None
Undefined reference	An element without a defined data type must have a reference. Manual correction: In the element property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of attribute	An element without a reference, a data type or a substitution group should have at least one attribute. Manual correction: Define an attribute for the element Automatic correction: None
Existence of particle	An element with an embedded complex type must contain child elements, groups, group particles and/or Any. Manual correction: Add items to complex element or delete complex element Automatic correction: None

Check	Description and Correction
Invalid cardinality	You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. This check is only available in a model targeted with XDR. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Group checks

The following XML model checks are made on groups:

Check	Description and Correction
Group name and code uniqueness	Group names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	A group without a name or a code must have a reference. Manual correction: In the group property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of particle	A group must contain elements, groups, group particles and/or Any. Manual correction: Add items to group or delete group Automatic correction: None

Check	Description and Correction
Invalid cardinality	You should define a minimum (0 or 1) and a maximum cardinality (1 or unbounded) for a group particle occurrence. This check is only available in a model targeted with XDR. Manual correction: Double-click the group particle symbol and type a value for Minimum (0 or 1) and Maximum (1 or unbounded) properties Automatic correction: None

Attribute checks

The following XML model checks are made on attributes:

Check	Description and Correction
Attribute name and code uniqueness	Attribute names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	An attribute without a name or a code must have a reference. Manual correction: In the attribute property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Undefined type	You must define a data type for an attribute. Manual correction: In the attribute property sheet, define a data type with the Type list or the Browse tool Automatic correction: None

Notation checks

The following XML model checks are made on notations:

Check	Description and Correction
Notation name and code uniqueness	Notations names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined notation	A notation must have at least one URI defined for Public or System properties. Manual correction: In the notation property sheet, define a URI in the Public or System boxes Automatic correction: None

Attribute group checks

The following XML model checks are made on attribute groups:

Check	Description and Correction
Attribute group name and code uniqueness	Attribute group names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	An attribute group without a name or a code must have a reference. Manual correction: In the attribute group property sheet, define a reference with the Reference list or the Browse tool Automatic correction: None
Existence of attributes	An attribute group must contain at least one attribute. Manual correction: Add attributes to attribute group or delete attribute group Automatic correction: Deletes unassigned attribute group

Import checks

The following XML model checks are made on imports:

Check	Description and Correction
Undefined schema location and namespace	An import must have at least a schema location or a namespace defined. Manual correction: In the import property sheet, define a URI for the schema location and/or the namespace. Example for a schema location: http://www.something.org/schemas/order.xsd Example for a namespace: http://www.something.-org/xml/ordering Automatic correction: None

Redefine checks

The following XML model checks are made on redefines:

Check	Description and Correction
Undefined schema location	You must define a schema location for a redefine. Manual correction: In the redefine property sheet, define a URI or select a schema file for the schema location. For example: customers.xsd Automatic correction: None
Existence of component	A redefine must contain at least one of the following items: simple type, complex type, group or attribute group. Manual correction: Add items to the redefine Automatic correction: None

Key checks

The following XML model checks are made on keys:

Check	Description and Correction
Key name and code uniqueness	Element names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code

Check	Description and Correction
Existence of fields	A key must contain at least one field. Manual correction: Add at least one field to the key or delete the key. For example: @numEmployee Automatic correction: Deletes unassigned key For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.
Undefined selector	You must define an XPath expression for a key selector attribute. Manual correction: In the key property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee Automatic correction: None For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

KeyRef checks

The following XML model checks are made on KeyRefs:

Check	Description and Correction
KeyRef name and code uniqueness	KeyRef names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Undefined reference	A keyRef must contain a reference to a key or a unique constraint. Manual correction: In the keyRef property sheet, define a reference to a key or a unique constraint with the Reference list Automatic correction: None

Check	Description and Correction
Existence of fields	A keyRef must contain at least one field. Manual correction: Add at least one field to the keyRef or delete the keyRef. For example: @numEmployee Automatic correction: Deletes unassigned keyRef For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.
Undefined selector	You must define an XPath expression for a keyRef selector attribute. Manual correction: In the keyRef property sheet, define an XPath expression for the selector attribute. For example: s:company/s:employee Automatic correction: None For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

Unique checks

The following XML model checks are made on uniques:

Check	Description and Correction
Unique name and code uniqueness	Unique constraint names and codes must be unique in the model. Manual correction: Modify the duplicate name/code Automatic correction: Appends a number to the duplicate name/code
Existence of fields	A unique constraint must contain at least one field. Manual correction: Add at least one field to the unique constraint or delete the unique constraint. For example: @numEmployee Automatic correction: Deletes unassigned unique constraint For more information on fields, see section Defining an identity constraint field in chapter Building an XML model.

Check	Description and Correction
Undefined Selector	You must define an XPath expression for a unique constraint selector attribute. Manual correction: In the unique constraint property sheet, define an XPath expression for the unique constraint selector attribute. For example: s:company/s:employee Automatic correction: None For more information on XPath expressions, see section Defining an identity constraint selector in chapter Building an XML model.

Extension checks

The following XML model checks are made on extensions:

Check	Description and Correction
Undefined base type	You must define a base type when you derive a complex type by extension. Manual correction: In the complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type list or the Browse tool Automatic correction: None

Restriction checks

The following XML model checks are made on restrictions:

Check	Description and Correction
Undefined base type	You must define a base type when you derive a simple or a complex type by restriction. Manual correction: In the simple or complex type property sheet, click the Properties tool beside the Derivation box to display the Extension property sheet and select a base type with the Base type list or the Browse tool Automatic correction: None
Existence of facet	A simple type restriction must have at least one facet defined. Facets are defined in the Detail, Enumerations and Patterns tabs of a simple type restriction property sheet. Manual correction: Define one or more facets in the simple type restriction property sheet Automatic correction: None

Simple type list checks

The following XML model checks are made on simple types:

Check	Description and Correction
Undefined base type	You must define a base type when you derive a simple type by list. Manual correction: In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type list property sheet and select a data type with the Type list or the Browse tool Automatic correction: None

Simple type union checks

The following XML model checks are made on simple type unions:

Check	Description and Correction
Undefined base type	You must define at least two data types when you derive a simple type by union. Manual correction: In the simple type property sheet, click the Properties tool beside the Derivation box to display the simple type union property sheet and type a white space separated list of at least two data types (qualified names) in the Member types box Automatic correction: None

Annotation checks

The following XML model checks are made on annotations:

Check	Description and Correction
Existence of items	An annotation must contain at least one URI for a Documentation or an Application Information. Manual correction: Define a URI for a Documentation or an Application Information Automatic correction: None

Manipulating XML Objects Graphically

The graphical interface of PowerDesigner allows you to manipulate XML objects within or between the Browser tree view and the diagram window.

A global object is right under the model item in the Browser tree view. It has no parent symbol in the diagram.

A local object is under a group particle item in the Browser tree view. It has a parent symbol in the diagram.

The following sections explain all the graphical operations allowed in PowerDesigner.

Local objects

You can move local objects within or between the Browser tree view and the diagram window.

Move
Select Tools ➤ General Options to make sure that Move is the Default action of the Drag & Drop option.

Within Browser

You can move a local object within the Browser to convert it into a global object. If the new global object does not appear in the diagram, select Symbol ➤ Show Symbols and click the corresponding tab to select the new global object

Within diagram

You can move a local object within the diagram to another group particle. It remains a local object, but with a new parent object.

You cannot move a local object within the diagram to convert it into a global object. It remains attached to its group particle.

From Browser to diagram

When you move a local object from the Browser to the diagram, a synonym is created, attached to the same group particle as the original symbol.

From diagram to Browser

You can move a local object from the diagram to the Browser, and convert it into a global object. If the new global object does not appear in the diagram, select Symbol ➤ Show Symbols and click the corresponding tab to select the new global object

Global objects

You can move global objects within or between the Browser tree view and the diagram window.

Move
Select Tools ➤ General Options to make sure that Move is the Default action of the Drag & Drop option.

Within Browser	You can move a global object within the Browser to convert it into a local object (under a group particle item), but you cannot move a global object within its own structure (as a child of itself).
Within diagram	You can move a global object within the diagram to convert it into a local object. Just move the global object symbol to a group particle symbol.
From Browser to diagram	When you move a global object from the Browser to the diagram, a synonym symbol is created in the diagram.
From diagram to Browser	You can move a global object from the diagram to the Browser, and convert it into a local object (under a group particle item). If the new local object does not appear in the diagram, double-click the Collapse node of the group particle symbol under which the former global object has been attached.

Example: converting a local object into a global object

The following procedure explains how to convert a local object into a global object.

❖ To convert a local object into a global object

1. Click a local object (child element or attribute) in the Browser or the diagram.

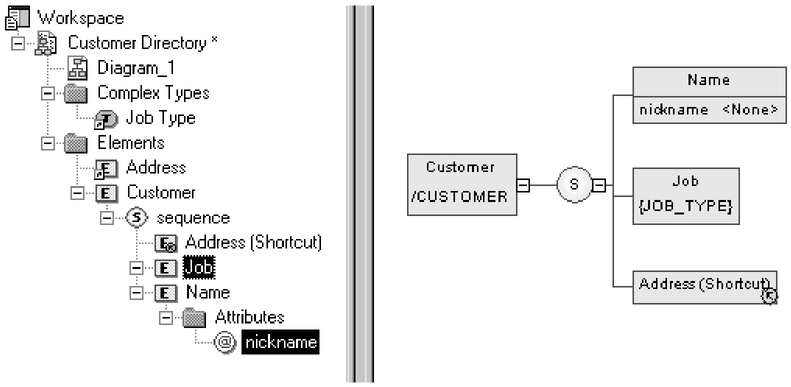
2. Drag and drop the local object in the space just beneath the Diagram item or the model item. A thick horizontal line indicates that you can drop the object.

   Drag and drop
   Select Tools ➤ General Options to make sure that Move is the Default action of the Drag & Drop option.

   The new global object is displayed in a global objects category (one level beneath the model item).

   If the new global object does not appear in the diagram, select Symbol ➤ Show Symbols and click the corresponding tab to select the object symbol.

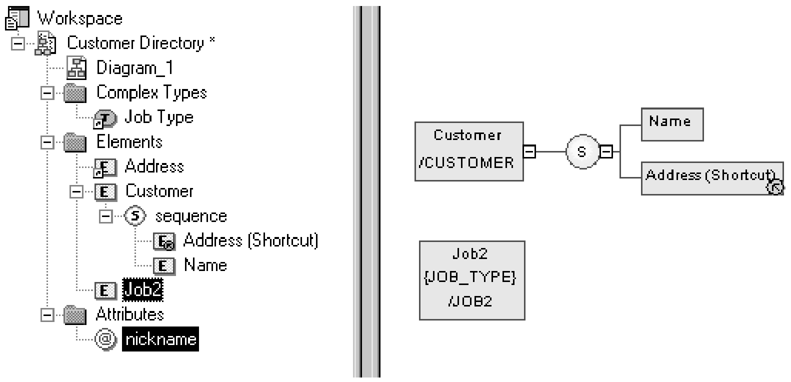
Example before local to global conversions within the Browser



Job is a child element of the Customer element.

Nickname is the attribute of the Name element.

After conversions



Job became Job2, a global element.

Nickname became a global attribute.

Comparing and Merging XML Models

You can compare and merge two XML models with the same XML language.

The comparison process allows you to highlight the differences between two XML models.

The merge process allows you to form a single model that combines design efforts performed independently by several team members.

Merge is performed from left to right, the model in the right pane is compared to the model in the left pane, differences are highlighted and merge actions are proposed in the model to be merged.

☞ For more information on comparing and merging models, see the Comparing and Merging Models chapter in the *Core Features Guide* .

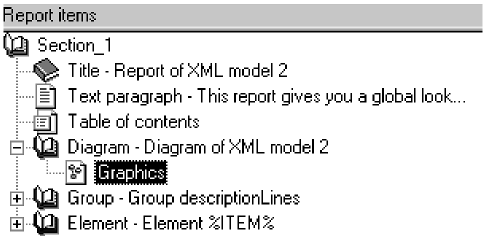
XML Model Reports

You can create reports to document your XML model.

☞ For detailed information about working with PowerDesigner reports, see the Reports chapter in the *Core Features Guide* .

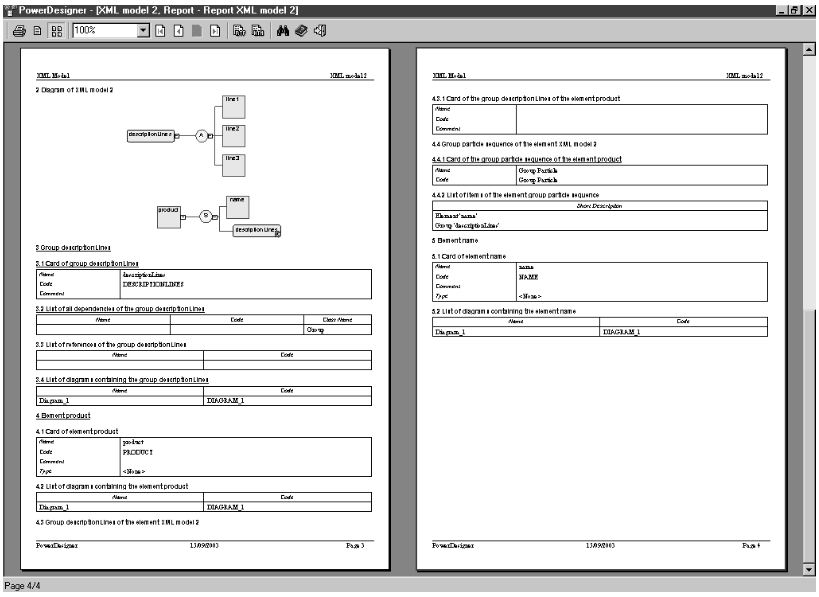
You can emphasize the hierarchical structure of your XML model by inserting diagram graphics in your report.

In the following structure, the graphical representation of the diagram will appear in the report just before the descriptive paragraphs.



Report items
- Section_1
  - Title - Report of XML model 2
  - Text paragraph - This report gives you a global look...
  - Table of contents
  - Diagram - Diagram of XML model 2
    - Graphics
  - Group - Group descriptionLines
  - Element - Element %ITEM%

To have a preview of your report, click the Print Preview tool in the Report Editor window.



PowerDesigner - [XML model 2, Report - Report XML model 2]

100%

XML Model — XML model 2

2 Diagram of XML model 2

3 Group descriptionLines

3.1 Card of group descriptionLines

Name	descriptionLines
Code	DESCRIPTIONLINES
Comment	

3.2 List of all dependencies of the group descriptionLines

Name	Code	Class Name
		Group

3.3 List of references of the group descriptionLines

Name	Code

3.4 List of diagrams containing the group descriptionLines

Name	Code
Diagram_1	DIAGRAM_1

4 Element product

4.1 Card of element product

Name	product
Code	PRODUCT
Comment	
Type	<None>

4.2 List of diagrams containing the element product

Name	Code
Diagram_1	DIAGRAM_1

4.3 Group descriptionLines of the element XML model 2

PowerDesigner 15/09/2003 Page 3

XML Model — XML model 2

4.3.1 Card of the group descriptionLines of the element product

Name	
Code	
Comment	

4.4 Group particle sequence of the element XML model 2

4.4.1 Card of the group particle sequence of the element product

Name	Group Particle
Code	Group Particle

4.4.2 List of items of the element group particle sequence

Short Description
Element 'name'
Group 'descriptionLines'

5 Element name

5.1 Card of element name

Name	name
Code	NAME
Comment	
Type	<None>

5.2 List of diagrams containing the element name

Name	Code
Diagram_1	DIAGRAM_1

PowerDesigner 15/09/2003 Page 4

Page 4/4

CHAPTER 4

Generating and Reverse Engineering XML Schemas

About this chapter

This chapter describes how to generate and reverse engineer an XML Schema Definition file (.XSD), a Document Type Definition file (.DTD) or an XML-Data Reduced file (.XDR).

The same procedures apply to all XML languages.

Contents

Generating an XML Schema

By default, PowerDesigner supports the generation of the following types of files for the XML schemas supported by the XSM:

Target Schema	Generated file
XML Schema Definition 1.0	XSD
Document Type Definition 1.0	DTD
XML-Data Reduced 1.0	XDR

You can preview the file to be generated by selecting the Preview tab of your XML model property sheet

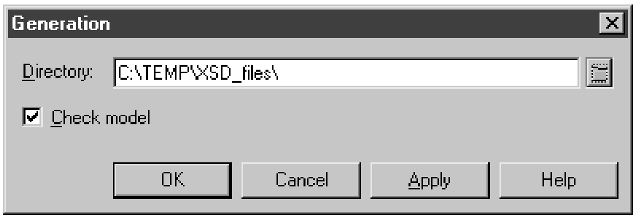
Parameter entities and DTD generation
Parameter entities are references to predefined values within a DTD file (See Parameter property in entity property sheet). During DTD generation, some object properties containing inadvertently parameter values will be generated with parameter references. If you are not satisfied with this default use of parameter entities, you should clear the Parameter property before generation.

Generating XML schema files

PowerDesigner provides a standard interface for generating all the supported XML schemas.

❖ To generate an XSD, DTD or XDR schema file

1. Select Language ➤ Generate *schema* File to open the Generation dialog:



Directory: C:\TEMP\XSD_files\

☑ Check model

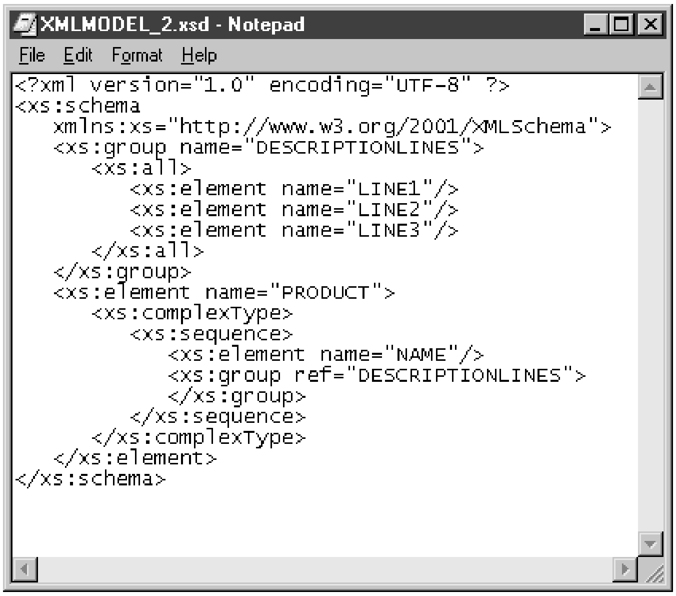
2. Enter a directory in which to generate the files and specify whether you want to perform a model check. For more information about checking your model, see “Checking an XSM” in the Working with XSMs chapter.

XDR generation options
When generating an XDR file, the Generation dialog contains an Options tab, where you can specify whether or not to generate comments (within a <description> tag). This option is enabled by default.

3. Click OK to begin generation.

   A Progress box is displayed. The Result list displays the files that you can edit. The result is also displayed in the Generation tab of the Output window, located in the bottom part of the main window.

4. Click Edit to edit the XSD, DTD or XDR file in your associated editor:



```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:group name="DESCRIPTIONLINES">
      <xs:all>
         <xs:element name="LINE1"/>
         <xs:element name="LINE2"/>
         <xs:element name="LINE3"/>
      </xs:all>
   </xs:group>
   <xs:element name="PRODUCT">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="NAME"/>
            <xs:group ref="DESCRIPTIONLINES">
            </xs:group>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Customizing XML schema generation

The PowerDesigner generation system is extremely customizable through the use of extended model definitions, profiles, and generation templates.

☞ For detailed information about customizing the generation of your code, including adding generation targets, options, and tasks, see *Customizing and Extending PowerDesigner.*

Reverse engineering an XML Schema into an XSM

Reverse engineering is the process of extracting an XML structure from an XML schema file, and using it to build or update an XSM.

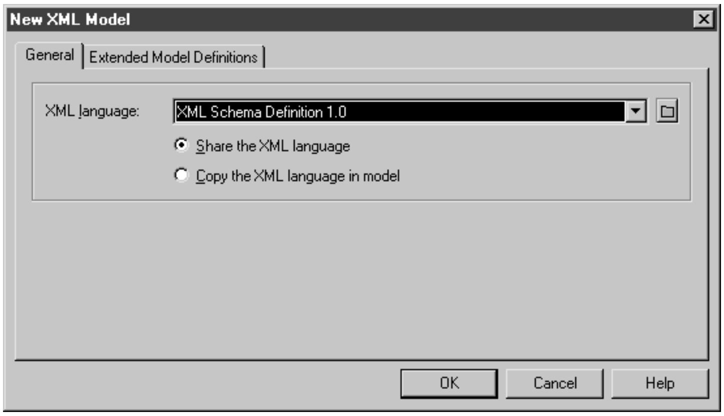
Parsing
PowerDesigner uses a parser software for XML reverse engineering, developed by the Apache Software Foundation (http://www.apache.org/).

Reverse engineering to a new XML model

You can reverse engineer XML schema files into a new XSM.

❖ To reverse engineer an XSD, a DTD or an XDR file to a new XSM

1. Select File ➤ Reverse Engineer ➤ XML Definition to open the New XML Model dialog box:



XML language: XML Schema Definition 1.0

(●) Share the XML language

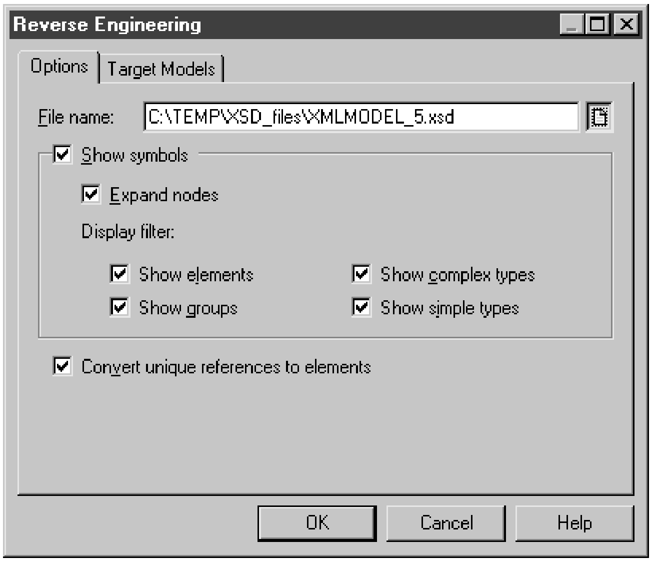
( ) Copy the XML language in model

2. Select an XML language and specify whether you want to share the resource file or copy it to your model.

3. [optional] Click the Extended Model Definitions tab, and select the extended model definitions you want to attach to the new model.

   ☞ For more information on extended model definitions, see “Extended Model Definitions” in the Resource Files and the Public Metamodel chapter of *Customizing and Extending PowerDesigner* .

4. Click OK to go to the Reverse Engineering dialog:


Reverse Engineering

Options | Target Models

File name: C:\TEMP\XSD_files\XMLMODEL_5.xsd

☑ Show symbols

☑ Expand nodes

Display filter:

☑ Show elements ☑ Show complex types

☑ Show groups ☑ Show simple types

☑ Convert unique references to elements

5. On the Options page, specify the file you want to reverse engineer, and select any appropriate options (see "Reverse engineering options" on page 127).

6. [optional] Click the Target Models tab and specify any existing PowerDesigner models which are referenced in the file being reverse engineered. These references will become shortcuts in the reversed model.

7. Click OK to begin reverse engineering.

   The XML file is reversed into an XML model and displayed in the diagram window and the Browser. The result is also displayed in the Reverse page of the Output window.

Reverse engineering options

The Options page of the Reverse Engineering dialog box displays the following options:

Option	Description
Show symbols	Creates symbols for the reversed objects in the diagram. If you select to show symbols, you can also specify to expand all the nodes, and to display elements, groups, and complex and simple types.
Convert unique references to elements	Enables the display of shortcuts to XML structures in other models as expandable nodes, instead of simple shortcuts. Since global objects with a single reference in the model will be converted into child objects, you should not use this option if you want to keep the global scope of some objects. You can subsequently perform this conversion by selecting Tools ➤ Convert Unique References in the XML model.

Reverse engineering to an existing XML model

You can reverse engineer XML schema files to add objects to an existing XSM.

❖ To reverse engineer an XSD, a DTD or an XDR file to an existing XSM

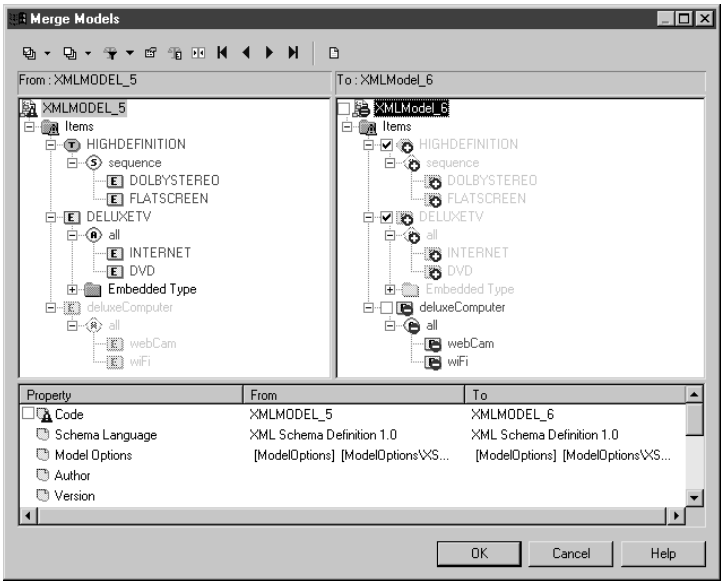
1. Open the XML model you want to reverse into and then select Language ➤ Reverse Engineer *schema* File to open the Reverse Engineering dialog box.

2. Select the file to reverse, and specify any appropriate options (see "Reverse engineering options" on page 127).

3. [optional] Click the Target Models tab and specify any existing PowerDesigner models which are referenced in the file being reverse engineered. These references will become shortcuts in the reversed model.

4. Click OK to begin reverse engineering.

   A message in the Output window confirms that the file has been reversed and the Merge Models window opens:



Property	From	To
Code	XMLMODEL_5	XMLMODEL_6
Schema Language	XML Schema Definition 1.0	XML Schema Definition 1.0
Model Options	[ModelOptions] [ModelOptions\XS...	[ModelOptions] [ModelOptions\XS...
Author		
Version		

5. Review the objects that you will be importing, and the changes that they will make to the model..

   For more information on merging models, see the Comparing and Merging Models chapter in the *Core Features Guide* .

6. Click OK to merge the selected changes into your model.

7. [optional] Select Symbol ➤ Auto-Layout to organize the new symbols in your diagram.

CHAPTER 5

Working with XML and Databases

About this chapter

This chapter describes how to use an XML model to store or retrieve data in databases supporting XML.

Contents

Introducing XML in Databases

Most relational databases now support XML so that you can store or retrieve data through XML files. You can use an XML model to generate an **annotated schema** that will allow you to store or retrieve data in such a database. The following databases are supported:

Database	Mapped XML model	Targeted XML language	Required XEM file
Microsoft SQL Server 2000	Yes	XSD or XDR	Microsoft SQL Server
Oracle 9i2	No	XSD	Oracle 9i2
IBM DB2 v8.1	Yes	DTD	IBM DB2 DAD

By attaching the SQL/XML extended model definition to an XML model mapped to a PDM, you can also generate **SQL/XML queries** to retrieve data in an XML format, from relational databases supporting SQL/XML.

The best way to generate SQL/XML queries is to use the XML Builder Wizard which helps you build an XML model from a PDM. The generated XML model is mapped to the PDM and automatically linked to the SQL/XML extended model definition. (See "Generating SQL/XML queries" on page 133)

Mapping database objects to an XML schema via the XML Builder Wizard

You can create an XSM to generate an annotated schema from a PDM via the XML Builder.

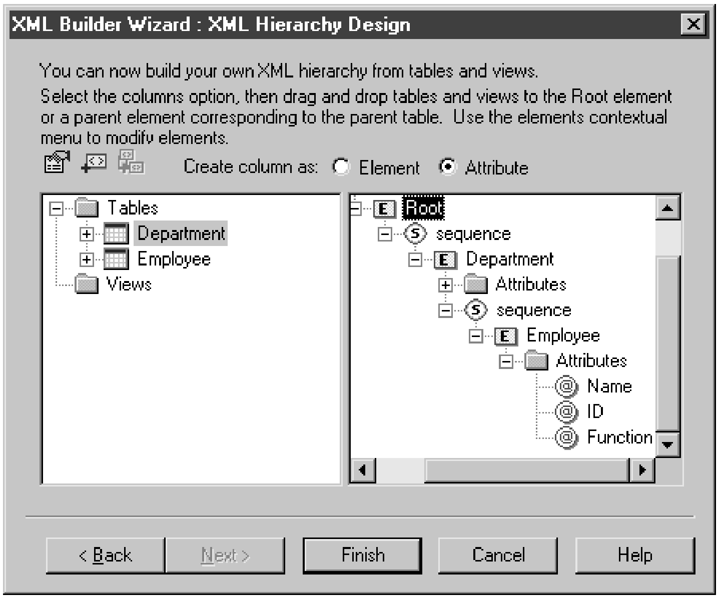
❖ To generate an annotated schema by mapping through the XML Builder Wizard

1. Open a PDM targeted with the appropriate DBMS, and select Tools ➤ XML Builder Wizard to open the XML Builder Wizard.

2. Specify whether to create a new or work with an existing model, and then click Next to go to the Tables and Views Selection page.

3. Select the tables and views from which you want to generate the schema, and then click Next to go to the XML Hierarchy Design page.

4. Build your hierarchy by dragging and dropping tables and/or columns from the left pane to the right pane or by using the tools above the panes:



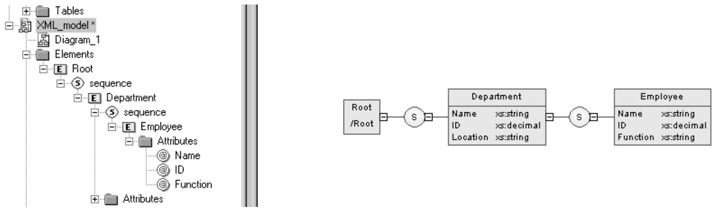
XML Builder Wizard : XML Hierarchy Design

You can now build your own XML hierarchy from tables and views.
Select the columns option, then drag and drop tables and views to the Root element or a parent element corresponding to the parent table. Use the elements contextual menu to modify elements.

Create column as: Element [✓ Attribute]

☞ For detailed information about using the wizard, see "Generating an XSM from a PDM via the XML Builder Wizard" in the Working with Data Models chapter of the *Data Modeling* guide.

5. Click Finish to generate the XML model:



In the case of an existing XML model, the generated elements appear alongside the existing elements.

Extended model definitions
The **SQL/XML** extended model definition is automatically attached to the generated XML model. You can attach the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema. (See Note in step 11)

Generating SQL/XML queries

SQL/XML is an XML extension of the Structured Query Language, which allows you to retrieve relational data using extended SQL syntax, and produce an XML result. SQL/XML has five main elements:

- **XMLELEMENT** - to edit an element with a name, a list of attributes (optional) and a list of values (optional)
- **XMLATTRIBUTES** - to edit a list of attributes with names and values
- **XMLAGG** - to edit in multiple rows a concatenation of elements, from a single XML value corresponding to a single column
- **XMLCONCAT** - to edit in the same row a concatenation of elements, from several XML values corresponding to several columns
- **XMLFOREST** - to edit in the same row a concatenation of elements, from several SQL values corresponding to several columns. The name and value of a column become the name and value of an element

An XML model allows you to generate SQL/XML queries for **global elements**, whatever the targeted XML language (XSD, DTD or XDR). You need first to map an XML model to a PDM, then to attach the SQL/XML extended model definition to the mapped XML model.

The best way to map an XML model to a PDM is to use the **XML Builder Wizard** from a PDM. The generated XML model is automatically mapped to the PDM and linked with the SQL/XML extended model definition. If need be, you can still modify the mapping through the Mapping tab of elements and complex types property sheets.

☞ For more information on the XML Builder Wizard, see Generating an XML model via the XML Builder Wizard, in the Working with Data Models chapter of the *Data Modeling* guide.

Generated SQL/XML queries cannot be parameterized.

Caution
The following procedure assumes you have an XML model open in the workspace and mapped to a PDM.

Attaching the SQL/XML extended model definition

The SQL/XML extended model definition allows you to generate SQL/XML queries. Note that this extended model definition is automatically linked to your XSM if you have created it from a PDM via the XML Builder Wizard

❖ To attach the XEM

1. Select Model ➤ Extended Model Definitions to open the List of Extended Model Definitions

2. Click the **Import an Extended Model Definition** tool to open the Extended Model Definition Selection dialog box, select SQL/XML on the General Purpose tab, and click OK in order to attach it to the model.

SQL/XML query preview
You can click the Preview tab of the model property sheet to obtain a preview of the SQL/XML query.

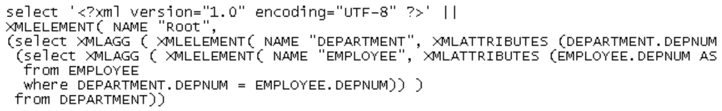
Generating an SQL/XML query file

♦ You can generate an SQL/XML query file from an XSM.

❖ To generate an SQL/XML query file

1. Select Tools ➤ Generate SQL/XML Queries to open the Generation dialog box.
2. Specify the directory in which to generate the file.
3. Click the Selection tab and specify which of the global elements you want to generate queries from. A separate file will be generated for each global element selected.
4. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the query file selected.
5. Click Edit to open the generated query file in your associated editor:



```
select '<?xml version="1.0" encoding="UTF-8" ?>' ||
XMLELEMENT( NAME "Root",
(select XMLAGG ( XMLELEMENT( NAME "DEPARTMENT", XMLATTRIBUTES (DEPARTMENT.DEPNUM
 (select XMLAGG ( XMLELEMENT( NAME "EMPLOYEE", XMLATTRIBUTES (EMPLOYEE.DEPNUM AS
  from EMPLOYEE
  where DEPARTMENT.DEPNUM = EMPLOYEE.DEPNUM)) )
 from DEPARTMENT))
```

Generating an Annotated Schema for Microsoft SQL Server 2000

Microsoft SQL Server 2000 is an XML-enabled database server, which supports annotations that can be used on XSD or XDR files, to map XML data to relational data.

An **annotated schema** is an XML file that allows you to store or retrieve data in an XML format, from relational databases supporting XML. An XML model allows you to generate an annotated schema (XSD or XDR) for SQL Server 2000.

❖ To generate an annotated schema for SQL Server 2000

1. Map an XSM to a PDM. You can do this manually or by generating an XSM from a PDM (or a PDM from an XSM) but we recommend that you use the XML Builder Wizard (see "Mapping database objects to an XML schema via the XML Builder Wizard" on page 132)
2. [if you do not use the wizard] Attach the SQL/XML extended model definition (see "Attaching the SQL/XML extended model definition" on page 134).
3. [optional] Reinforce the mappings with extended attributes (see "Reinforcing SQL Server mappings with extended attributes" on page 136).
4. Generate the annotated schema (see "Generating the SQL Server annotated schema file" on page 139).

Reinforcing SQL Server mappings with extended attributes

The Microsoft SQL Server extended model definition allows you to modify or reinforce the mapping resulting from the XML Builder Wizard via extended attributes.

If element and attribute names match table and column names

If the element and attribute names match the table and column names, you do not need to define extended attributes for XML objects.

❖ To attach the XEM

1. Select Model ➤ Extended Model Definitions to open the List of Extended Model Definitions

2. Click the **Import an Extended Model Definition** tool to open the Extended Model Definition Selection dialog box, click the **XML in Database** tab, select Microsoft SQL Server, and click OK in order to attach it to the model.

Annotated schema preview
By clicking the Preview tab of the model property sheet, you can have a preview of the annotated schema.

SQL Server extended attributes for elements and attributes

You can set extended attributes on elements and attributes to reinforce their mapping to tables and columns, by opening their property sheets and clicking the Extended Attributes tab.

Annotation	Description
encode	When an XML element or attribute is mapped to a SQL Server BLOB column, allows requesting a reference (URI) to be returned and used later to return BLOB data. Available for: Element, Attribute
field	Maps an XML item to a database column. Available for: Element, Attribute
hide	Hides the element or attribute specified in the schema in the resulting XML document. Available for: Element, Attribute
is-constant	Creates an XML element that does not map to any table. The element is displayed in the query output. Available for: Element
key-fields	Allows specification of columns that uniquely identify the rows in a table. Available for: Element
limit-field	Allows limiting the values that are returned on the basis of a limiting value. Available for: Element, Attribute

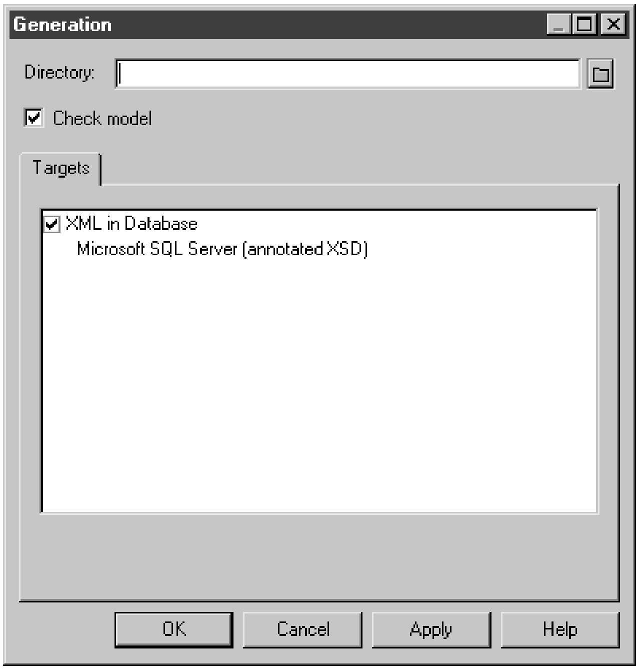
Annotation	Description
limit-value	Allows limiting the values that are returned on the basis of a limiting value. Available for: Element, Attribute
mapped	Allows schema items to be excluded from the result. Available for: Element, Attribute
max-depth	Allows you to specify depth in recursive relationships that are specified in the schema. Available for: Element
overflow-field	Identifies the database column that contains the overflow data. Available for: Element
relation	Maps an XML item to a database table. Available for: Element
relationship-child	Specifies an element as the **child table** in a reference (To define only in the child element property sheet). Available for: Element
relationship-child-key	Specifies an attribute as the **foreign key** of a child table in a reference (To define only in the child element property sheet). Available for: Element
relationship-parent	Specifies an element as the **parent table** in a reference (To define only in the child element property sheet). Available for: Element
relationship-parent-key	Specifies an attribute as the **primary key** of a parent table in a reference (To define only in the child element property sheet). Available for: Element
use-cdata	Allows specifying CDATA sections to be used for certain elements in the XML document. Available for: Element
prefix	Creates valid XML ID, IDREF, and IDREFS. Prepends the values of ID, IDREF, and IDREFS with a string. Available for: Attribute

Generating the SQL Server annotated schema file

You generate the annotated schema file by selecting it as an additional target for standard schema generation.

❖ To generate an annotated schema file

1. Select Language ➤ *schema* File to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



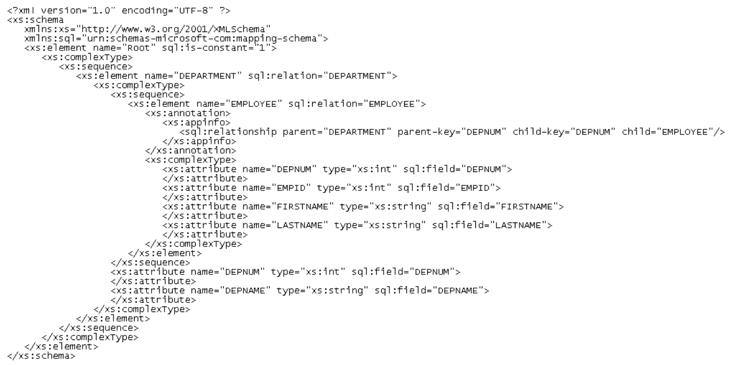
Directory:

☑ Check model

☑ XML in Database
Microsoft SQL Server (annotated XSD)

3. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the annotated schema file selected.
4. Click Edit to open the generated annotated schema in your associated editor:



```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
   <xs:element name="Root" sql:is-constant="1">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="DEPARTMENT" sql:relation="DEPARTMENT">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="EMPLOYEE" sql:relation="EMPLOYEE">
                        <xs:annotation>
                           <xs:appinfo>
                              <sql:relationship parent="DEPARTMENT" parent-key="DEPNUM" child-key="DEPNUM" child="EMPLOYEE"/>
                           </xs:appinfo>
                        </xs:annotation>
                        <xs:complexType>
                           <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
                           </xs:attribute>
                           <xs:attribute name="EMPID" type="xs:int" sql:field="EMPID">
                           </xs:attribute>
                           <xs:attribute name="FIRSTNAME" type="xs:string" sql:field="FIRSTNAME">
                           </xs:attribute>
                           <xs:attribute name="LASTNAME" type="xs:string" sql:field="LASTNAME">
                           </xs:attribute>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="DEPNUM" type="xs:int" sql:field="DEPNUM">
                  </xs:attribute>
                  <xs:attribute name="DEPNAME" type="xs:string" sql:field="DEPNAME">
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Note the SQL namespace (with the sql prefix) and the SQL annotations for tables (sql:relation), columns (sql:field) and reference (sql:relationship).

Generating an Annotated Schema for Oracle 9i2

Oracle 9i2 is a database server with a native XML storage and retrieval technology called Oracle XML DB. There is no mapping between XML data and relational data. Tables, columns and abstract data types (ADT) are created from **annotated schemas** (XSDs). Annotated schemas are XML-coded files, targeted with an XML language and tagged with specific DBMS annotations, that allow you to store or retrieve data in an XML format, from relational databases supporting XML.

An XML model allows you to generate an annotated schema (XSD) for Oracle 9i2. You just need to attach the Oracle 9i2 **extended model definition** to the XML model. Oracle 9i2 uses by default the name of the XML elements present in the annotated schema to generate SQL objects. You can override the creation of SQL objects by defining **extended attributes** for elements, complex types and the XML model.

Caution
The following procedure assumes you have an XML model open in the workspace and targeted with XSD.

Creating Oracle mappings with extended attributes

The Oracle XML DB extended model definition allows you to set extended attributes to define mappings between your XSM and an Oracle database.

❖ To attach the XEM

1. Select Model ➤ Extended Model Definitions to open the List of Extended Model Definitions
2. Click the **Import an Extended Model Definition** tool to open the Extended Model Definition Selection dialog box, click the **XML in Database** tab, select Oracle XML DB, and click OK in order to attach it to the model.

Oracle extended attributes for elements and attributes

You can set extended attributes on various XSM objects to define mappings with an Oracle database.

Element extended attributes

The following annotations can be specified on the Extended Attributes tab of the property sheets of elements:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
columnProps	Specifies the column storage clause that is inserted into the default CREATE TABLE statement. It is useful mainly for elements that are mapped to tables, namely top-level element declarations and out-of-line element declarations
defaultTable	Specifies the name of the table into which XML instances of this schema should be stored. This is most useful in cases when the XML is being inserted from APIs where table name is not specified (for example, FTP and HTTP)
javaClassname	Used to specify the name of a Java class that is derived from the corresponding bean class, to ensure that an object of this class is instantiated during bean access. If a JavaClassname is not specified, Oracle XML DB will instantiate an object of the bean class directly
maintainDOM	If true, instances of this element are stored so that they retain DOM fidelity on output. This implies that all comments, processing instructions, namespace declarations, and so on, are retained in addition to the ordering of elements. If false, the output need not be guaranteed to have the same DOM behavior as the input
maintainOrder	If true, the collection is mapped to a VARRAY. If false, the collection is mapped to a NESTED TABLE
SQL-CollSchema	Name of the database user owning the type specified by SQLCollType
SQLCollType	Specifies the name of the SQL collection type corresponding to this XML element that has maxOccurs $>$ 1

Annotation	Description
SQLInline	If true this element is stored inline as an embedded attribute (or a collection if maxOccurs > 1). If false, a REF (or collection of REFs if maxOccurs > 1) is stored. This attribute will be forced to false in certain situations (like cyclic references) where SQL will not support inlining
SQLName	Specifies the name of the attribute within the SQL object that maps to this XML element
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration
tableProps	Specifies the TABLE storage clause that is appended to the default CREATE TABLE statement. This is meaningful mainly for global and out-of-line elements

Complex type extended attributes

The following annotations can be specified on the Extended Attributes tab of the property sheets of complex types:

Annotation	Description
beanClassname	Can be used within element declarations. If the element is based on a global complexType, this name must be identical to the beanClassname value within the complexType declaration. If a name is specified by the user, the bean generation will generate a bean class with this name, instead of generating a name from the element name
SQLSchema	Name of the database user owning the type specified by SQLType
SQLType	Specifies the name of the SQL type corresponding to this XML element declaration

Model extended attributes

The following annotations can be specified on the Extended Attributes tab of the property sheet of the model:

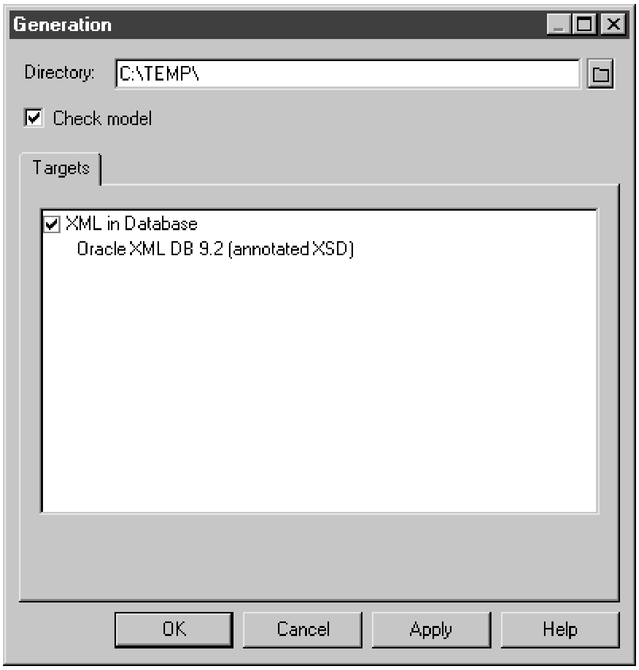
Annotation	Description
mapUnboundedString-ToLob	If true, unbounded strings are mapped to CLOB by default. Similarly, unbounded binary data get mapped to BLOB, by default. If false, unbounded strings are mapped to VARCHAR2(4000), and unbounded binary components are mapped to RAW(2000)
storeVarrayAsTable	If true, the VARRAY is stored as a table (OCT). If false, the VARRAY is stored in a LOB

Generating the Oracle annotated schema file

You generate the annotated schema file by selecting it as an additional target for standard schema generation.

❖ To generate an annotated schema file

1. Select Language ➤ *schema* File to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



Directory: C:\TEMP\

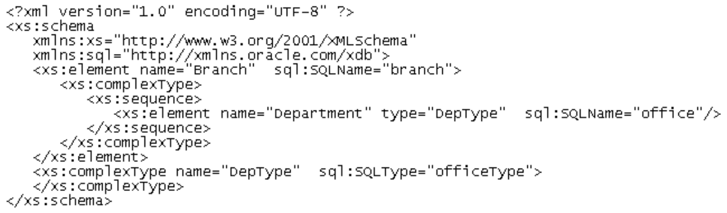
☑ Check model

☑ XML in Database
Oracle XML DB 9.2 (annotated XSD)

3. Click OK to begin the generation.

   The Result dialog box is displayed with the path of the annotated schema file selected.

4. Click Edit to open the generated annotated schema in your associated editor:



```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:sql="http://xmlns.oracle.com/xdb">
   <xs:element name="Branch"  sql:SQLName="branch">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Department" type="DepType"  sql:SQLName="office"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="DepType"  sql:SQLType="officeType">
   </xs:complexType>
</xs:schema>
```

   Note the Oracle namespace (with the sql prefix) and annotations for tables (sql:SQLName) and ADTs (sql:SQLType)

Generating a DAD File for IBM DB2

IBM DB2 v8.1 (or higher) is a database server with an add-in for XML storage and retrieval called IBM DB2 Extender. XML data (elements, attributes) are mapped to relational data (tables, columns) through Document Access Definition files (.DAD).

There are three types of DAD files:

Storage Type	Description
Xcolumn	Column mapping - the Root element is mapped to a table, and its attributes or child elements are mapped to columns identified by an XPath
Xcollection	SQL mapping - the DAD file starts with a SQL statement for the table mapped to the Root element, and each child element or attribute is mapped to a column or a table name
Xcollection	RDB mapping - a Relational Database node, with a table and a column name, is associated with each attribute or child element of the Root element

An XML model targeted with DTD allows you to generate DAD files for IBM DB2.

❖ To generate an DAD files for DB2

1. Map an XSM to a PDM. You can do this manually or by generating an XSM from a PDM (or a PDM from an XSM) but we recommend that you use the XML Builder Wizard (see “Mapping database objects to an XML schema via the XML Builder Wizard” on page 132)
2. [if you do not use the wizard] Attach the SQL/XML extended model definition (see “Attaching the SQL/XML extended model definition” on page 134).
3. Further specify the mappings with extended attributes (see “Reinforcing DB2 mappings with extended attributes” on page 148).
4. Generate the annotated schema (see “Generating a DB2 DAD file” on page 149).

Mapping DB2 objects to an XML schema via the XML Builder Wizard

You can generate an annotated schema via the XML Builder.

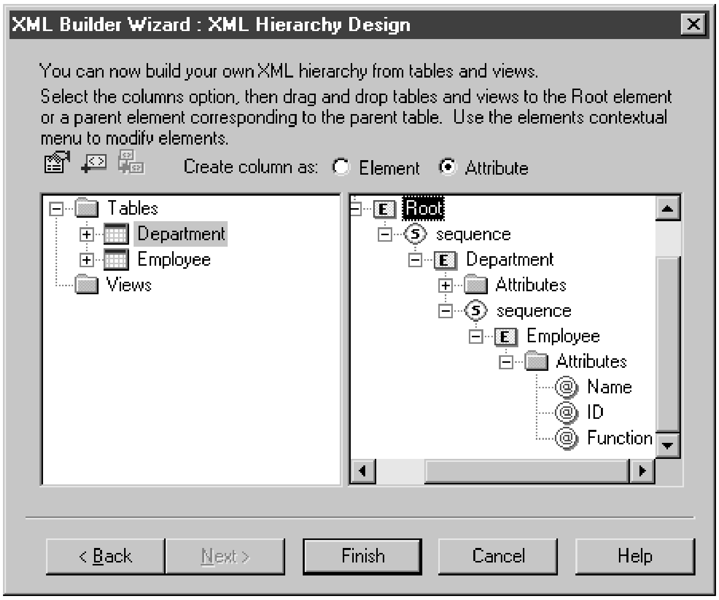
❖ **To generate an annotated schema by mapping through the XML Builder Wizard**

1. Open a PDM targeted with the IBM DB2 UDB 8.x Common Server DBMS, and select Tools ➤ XML Builder Wizard to open the XML Builder Wizard.

2. Specify whether to create a new or work with an existing model, and then click Next to go to the Tables and Views Selection page.

3. Select the tables and views from which you want to generate the schema, and then click Next to go to the XML Hierarchy Design page.

4. Build your hierarchy by dragging and dropping tables and/or columns from the left pane to the right pane or by using the tools above the panes:



XML Builder Wizard : XML Hierarchy Design

You can now build your own XML hierarchy from tables and views.
Select the columns option, then drag and drop tables and views to the Root element or a parent element corresponding to the parent table. Use the elements contextual menu to modify elements.

Create column as: ○ Element [✓ Attribute]

☞ For detailed information about using the wizard, see “Generating an XSM from a PDM via the XML Builder Wizard” in the Working with Data Models chapter of the *Data Modeling* guide.

5. Click Finish to generate the XML model:

   In the case of an existing XML model, the generated elements appear alongside the existing elements.

Extended model definitions
The **SQL/XML** extended model definition is automatically attached to the generated XML model. You can attach the **XML Document** extended model definition to generate a simplified XML file that will help you understand the annotated schema.

Reinforcing DB2 mappings with extended attributes

The IBM DB2 DAD extended model definition allows you to modify or reinforce the mapping resulting from the XML Builder Wizard via extended attributes.

❖ To attach the XEM

1. Select Model ➤ Extended Model Definitions to open the List of Extended Model Definitions
2. Click the **Import an Extended Model Definition** tool to open the Extended Model Definition Selection dialog box, click the **XML in Database** tab, select IBM DB2 DAD, and click OK in order to attach it to the model.

DAD file preview
In the Preview tab of the Root element property sheet, click the **DB2XMLExtender.DAD File** tab to preview the DAD file. If the DAD File tab is not available, click the Select Generation Targets tool to select IBM DB2 DAD in the Targets list and click OK.

DB2 extended attributes for global elements

You can set extended attributes on global elements to reinforce their mapping to tables and columns, by opening their property sheets and clicking the Extended Attributes tab.

Extended attribute	Description
Database	Name of the database
DTDID	ID added to the DTD_ref system table in DB2 XML Extender
Login	Name of the logged-in user
MappingType	Type of mapping for a collection
NamespaceNode	Text zone where each line describes a namespace couple (name = value). The separator character is '='

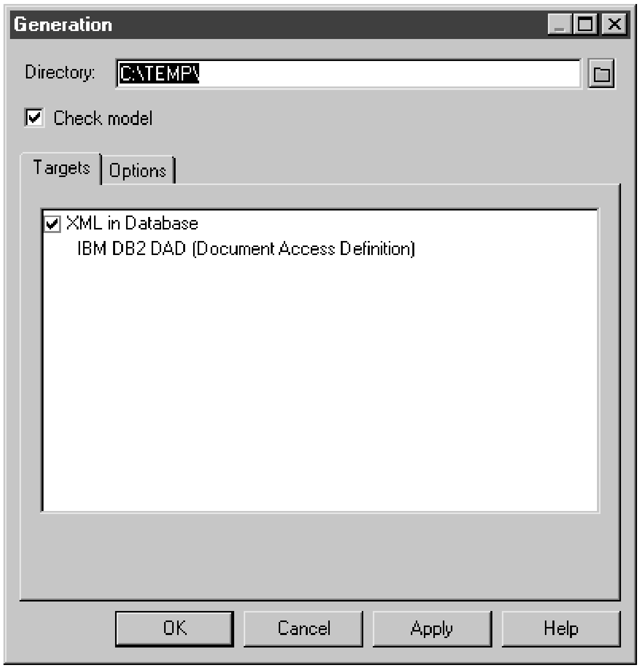
Extended attribute	Description
Password	Password of the logged-in user
PathGeneration	Generation path
ProcessInstruction	A text zone that enables the user to enter some instruction
SideTableID	Identifier of the side table (optional)
SideTableName	Name of the side table
StorageName	If StorageType is Xcolumn, then it is the name of the sidetable column
StorageType	Type of storage (Xcollection or Xcolumn)

Generating a DB2 DAD file

You generate the DAD file by selecting it as an additional target for standard schema generation.

❖ To generate an annotated schema file

1. Select Language ➤ *schema* File to open the Generation dialog box.
2. Specify the directory in which to generate the file and select the XML in Database target on the Targets tab.



☑ XML in Database
IBM DB2 DAD (Document Access Definition)

3. [optional] Click the Options tab, and set any appropriate generation options (see “DB2 DAD file generation options” on page 151).

4. Click OK to begin the generation.

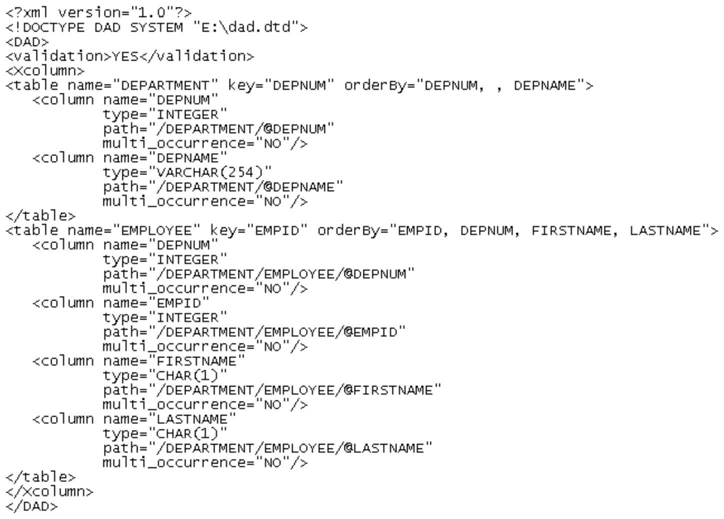
   The Result dialog box is displayed with the path of the generated DAD, DTD and SQL files.

5. Click Edit to open the generated DAD file in your associated editor:

   ♦ Extract of a DAD file defined with **Xcollection** as StorageType, and **RDB** as MappingType:

```
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Root SYSTEM "C:\TEMP\DTD_files\CorporateMembership.dtd"</doctype>
<root_node>
<element_node name="Root">
   <element_node name="DEPARTMENT">
      <attribute_node name="DEPNUM">
         <RDB_node>
            <table name="DEPARTMENT"/>
            <column name="DEPNUM" type="INTEGER"/>
         </RDB_node>
      </attribute_node>
      <attribute_node name="DEPNAME">
         <RDB_node>
            <table name="DEPARTMENT"/>
            <column name="DEPNAME" type="VARCHAR(254)"/>
         </RDB_node>
      </attribute_node>
      <element_node name="EMPLOYEE">
         <attribute_node name="DEPNUM">
            <RDB_node>
               <table name="EMPLOYEE"/>
               <column name="DEPNUM" type="INTEGER"/>
            </RDB_node>
         </attribute_node>
```

♦ DAD file defined with **Xcolumn** as StorageType:



```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "E:\dad.dtd">
<DAD>
<validation>YES</validation>
<Xcolumn>
<table name="DEPARTMENT" key="DEPNUM" orderBy="DEPNUM, , DEPNAME">
   <column name="DEPNUM"
           type="INTEGER"
           path="/DEPARTMENT/@DEPNUM"
           multi_occurrence="NO"/>
   <column name="DEPNAME"
           type="VARCHAR(254)"
           path="/DEPARTMENT/@DEPNAME"
           multi_occurrence="NO"/>
</table>
<table name="EMPLOYEE" key="EMPID" orderBy="EMPID, DEPNUM, FIRSTNAME, LASTNAME">
   <column name="DEPNUM"
           type="INTEGER"
           path="/DEPARTMENT/EMPLOYEE/@DEPNUM"
           multi_occurrence="NO"/>
   <column name="EMPID"
           type="INTEGER"
           path="/DEPARTMENT/EMPLOYEE/@EMPID"
           multi_occurrence="NO"/>
   <column name="FIRSTNAME"
           type="CHAR(1)"
           path="/DEPARTMENT/EMPLOYEE/@FIRSTNAME"
           multi_occurrence="NO"/>
   <column name="LASTNAME"
           type="CHAR(1)"
           path="/DEPARTMENT/EMPLOYEE/@LASTNAME"
           multi_occurrence="NO"/>
</table>
</Xcolumn>
</DAD>
```

DB2 DAD file generation options

The following generation options are available:

Option	Description
Character ending an instruction	Character ending instructions in the SQL file for stored procedures
Generates procedures deployment	Generation of a SQL script for stored procedures enabling XML data storage and facilitating XML data retrieval
Path of DAD.dtd	Path of the DTD file installed with IBM DB2 Extender and describing the specific syntax of DAD files
Schema validation	Validation tag in the DAD files to check the conformity of DAD files with the DAD syntax

Index